SINCLAIR QL WORLD

OCTOBER 1991 £1.95

M.C.M
Quality Editorial

9 770951 933016 10

A QL cousin:
The One-Per-Desk

Software File:
The Painter
update

Get inside your QL's head:
SYSTEMATIC MACHINE CODE PROGRAMMING

SINCLAIR QL WORLD

Editor
Helen Armstrong

Production Controller
Jayne Penfold

Designer
Jeff Gurney

Advertising Manager
Jason Newman

Magazine Services
Yvonne Taylor

Advertising Production
Michelle Evans

Group Advertising Manager
Lynda Elliott

Group Editor
John Taylor

Publishing Director
Wendy Palmer

Group Deputy Managing Director
Ray Lewis

Group Managing Director
Peter Welham

Sinclair QL World
Panini House
116-120 Goswell Road
London EC1V 7QD
Telephone 071-490 7161
ISSN 026806X

Unfortunately, we are no longer able to answer enquiries made by telephone. If you have any comments or difficulties, please write the The Editor, Open Channel, Trouble Shooter, or Psion Solutions. We will do our best to deal with your problem in the magazine, though we cannot guarantee individual replies.
Back issues are available from the publisher price £2 U.K., £2.75 Europe. Overseas rates on request.
Published by Maxwell Consumer Magazines, A Division of MCPC Ltd., Sinclair QL World is distributed by IPC Market force, King's Reach Tower, Stamford Street, London SE1 9LS.
Subscription information from: MCM Subscription Dept, Lazahold Ltd., PO Box 10, Roper St, Pallion Ind. Est., Sunderland SR4 4SN. Tel: 091 510 2290
UK: £21.00, Europe: £32.00,
Rest of the World: £38.00.

Typesetting by Ford Graphics, 8-10 Whitsbury Road, Fordingbridge, Hampshire. SP6 1BR. Tel: (0425) 655657
Printed and bound by BPCC, Colchester. Covers printed by Spottiswoode Ballantyne, Colchester.
Sinclair QL World is published on the third Thursday preceding cover date.

CONTENTS

OCTOBER 1991

NEXT MONTH

PROSPERO PASCAL
An alternative language for the portable programmer.

ONE MAN'S SYSTEM
Getting into the swim with Archive.

TROUBLE A PROBLE

Brian Davies reflects on screen shots and emulators.

The process of transferring files from floppy to hard disk caused caused me a temporary loss of *text*[87], which did not want to run from the hard disk. The RECON routine had been run, and the device changed to WIN1_, but the program wasn't getting past the start of displaying it as text area when loaded. The way out of the problem was provided by software[87], with the suggestion to try deleting the file CONFIGURE_C87 from the hard disk. When this had been done, the program started normally and it was possible then to alter the Text and Storage Device settings to WIN1_ and save a new CONFIGURE_C87 file.

While my reservations about the implementation of the directory structure on the Miracle hard disk still remain, there is no doubt that the space and speed brought with it by the hard disk make QL life much more pleasant. The freedom to be able to copy a seemingly endless string of large files onto the hard disk, without coming near running out of space, relieves a considerable pressure from the user who has been having to consider every move very carefully previously.

What's your rom?

Do you know what version your Qdos rom chips are? (Type PRINT VER$, then enter, and the version will be printed on the screen.)

Many users possibly do not have the faintest idea of their version, and are not really interested. In some senses unfortunately, programmers *are* interested, and they tend to use the 'best' version – JS. This is only to be expected; if the JS solves some faults in the computer, it is reasonable to use it. What is sometimes forgotten though, is that many users do not have this version. The result may be that the program, which works fine for the programmer, causes trouble for the user. This has happened with three programs I have used recently, and two of them were effectively unusable on my JM. My feeling about this is rather the same as with cars – petrol should be designed to suit the car, not the car to suit the petrol, and it is distinctly upsetting to have a relatively expensive investment rendered almost useless by changes in its diet. Can't programmers test on more than one version of rom? QLs are not that expensive.

Version 2.08 of *Discopy* from Qfile is now out, with a change having been made to the boot procedure to prevent the 'in use' message, noted on a JM QL when version 2.06 was reviewed.

Printing

Once you get a new printer, you find all sorts of things to try it out on. The higher-quality output of ink jet and laser printers (compared to dot-matrix types) is clear to see when text is printed from a word-processing program, but the advantage is less obvious to see when printing graphics, from a DTP program for instance. Prints on a Hewlett-Packard DeskJet and Epson GQ-5000 laser from *Professional Publisher* seem better in appearance, but mainly because the paper gets less damaged by the print process (no impact from pins) and these printers use cut-sheet paper, which may be of better surface appearance than the continuous paper often used in DMP printers. So the quality of the image is essentially no better, being determined, to a large extent, by the printer-driver and printer-emulation mode used, which has to be 'Epson-compatible' (that is, FX80-compatible, a rather ancient and basic standard for both text and graphics). If my figures are correct, the resolution of the laser in its native mode is about four times better than that of the FX80.

The DeskJet (using an FX80 emulation cartridge) also produced some oddities in text. For example, lots of 's' characters looked more like 'z'. The laser won't (so far) print a full page on one sheet of paper; it spreads one ProPub page over three sheets. This problem is somewhat mystifying, and comments on it would be appreciated. The impression is given (by the printer) that it has run out of memory, part-way through a print. It says 'buffer full'. Bearing in mind that the page being printed is on a disk file of less than 100 KB, and the same printer happily handles files of around 500 KB from other sources, lack of memory *per se* can't be the problem. The printer has 2 MB of ram anyway. What does seem to be involved is the input buffer of the printer, which can be no larger than 51 KB.

In the picture

The quality of photographic printing from shops and mail-order companies is nothing to boast about, but it usually seems that the negatives and transparencies are good enough; it's the prints that are poor, possibly mainly because of the inadequacies of automatic focusing devices used in the print process. If you want to take shots of your QL screen, you don't really have to worry about your camera being a cheap one, since the advantages of a good camera tend to get thrown away by the printers. Read the part of the *QL User Guide* devoted to Easel for advice. Basically, a shutter speed of no less than about 1/4 second is advised for TV screens. This is to ensure that an even, steady picture is obtained. The TV screen picture takes about 1/25 of a second to 'draw', so the exposure needs to be greater than this. With a 100 ASA film, an aperture of f5.6 is recommended. To reduce screen distortion, it is preferable to use a lens of 100 mm or so focal length.

My own recent shots of a colour monitor screen were taken using cheap, mail-order 100 ASA film, a shutter speed of 1/5 or 1/2 second, an aperture of f5.6 and a 135 mm lens. On reflection, the shutter speed should have been higher, say, 1/15 second, but my camera offers only 1/25 in this range. That is rather too close to the screen 'painting' rate, but the shutter may have got a bit slow in its old age, and could in practice be about 1/20 and suitable for the job. The 1/3-second exposures were wasted. The telephoto lens does necessitate the camera being 2 metres away from the screen, but it reduces distortion. For one display, the camera tripod was backed against a cupboard door, and it was essential to have a viewfinder which allowed looking down on the image, since there was no room to get behind the camera and look through the usual viewfinder. Use a cable release for the shutter, to reduce the risk of shake. Make sure the room is dark, and that there are no reflections on the screen. Adjust the display controls to make the image sharp and free of glare, but make sure no important colours are lost by turning brightness of contrast controls too far.

Boots charge £4.59 to do one-day processing of 24-exposure 35 mm film – not cheap, but acceptable if you are in a hurry. The prints must be glossy for the printers who have to transfer the image to

the printed page. Don't bank on prints coming out well. The last set of mine had several chopped at the edges, and out-of-focus, despite the transparencies looking quite reasonable. The transparencies themselves are likely to be preferable, anyway, because they are much sharper and allow the printers more flexibility. While we do use black and white screen dumps (a graphics transfer of the screen image to a printer) to illustrate program reviews, and they are satisfactory in many cases, colour can be important with some programs and it does look more attractive on a magazine page.

Brief encounter: Atari QL emulator

QL World has been asking for owners of an Atari ST with the QL emulator fitted to step forward and let the rest of us know something about this combination. One (very satisfied) owner of the combination is Alf Kendall, and he was kind enough to bring the system round and demonstrate it to me. Here are a few notes based on watching the system demonstrated for most of one day.

The basic computer is an Atari Mega 2, with 2 MB (expandable to 4 MB) of ram memory and an 8 MHz 16-bit 68008 processor chip, and one 3½in floppy drive.

The computer casing is about the same width as the Phillips CM 8833 colour monitor which sits on top of it. There is another similar-sized box underneath, in which resides a 30 MB hard disc drive. There is room for another hard disc, and – with some persuasion – a floppy drive, but these are not presently fitted to Alf's system. He has another 3½in drive attached externally. While the units mentioned so far make up quite a useful system, there are a trio of extra hardware items which make it really interesting.

An accelerator board boosts the CPU speed to 16 MHz. Two other internally-mounted boards give the capability to emulate MS-DOS – that is, a PC – and Qdos – a QL. So, you effectively have three computers in one. You can also buy a Macintosh emulator, to make that four computers. It may sound a bit of a cobbled-up thing, but it really does work, and work well. The hard disk (on Alf's system) is partitioned into three sections: Atari dos, MS-DOS and Qdos. The MS-DOS emulator says it has a '16-bit 8 MHz 80286' CPU, and it certainly behaves as though that's true. It appears to run PC programs at a similar speed to my 10 MHz 286 machine, and the disk-access speed is nearly as high as on my machine (which has a 0.5 MB disk cache to assist it). Using the Norton Utilities System Information benchmark routine, the overall speed was about eight times that of the reference PC/XT. The two floppy drives and one partition of the hard drive are usable with MS-DOS. Various standard programs (eg *WordPerfect 5.0, Flight Simulator 3.0, Lotus Manuscript, Psion Abacus* and *Quill)* ran without a problem. The screen display was of only CGA standard, but it does look likely that there is a way of obtaining EGA; this is something to be checked later.

The Qdos emulator has been around for three to four years and is well debugged. The only program Alf quoted as not running properly is the configuration routine for *Turbo-Plus-Quill.* We ran *Taskmaster, Flashback, Perfection, text*^{87}*, Archive, Professional Publisher* and various other standard QL programs. The absence of trouble was impressive. Equally impressive was the speed, generally quoted as about three times that of basic QL, for the basic 8 MHz Atari. Even Quill ceases to be a real drag on this combination computer. Not surprisingly, Perfection zooms along. Going up and down on a ProPub page with the cursor checked out as being six times as fast as on my JM QL, and the screen movement was much smoother. The speed- up factor is very likely overall more like six than three because the 16 Mhz accelerator board improves QL performance as well as Atari (but apparently doesn't do anything for the MS-DOS emulator). Alf uses a small routine which allows him to select different sub-directories, before switching (with Taskmaster) from one program to another. This deals with the problem experienced with the Miracle hard drive on the QL, where you can find the program you have switched to working from the wrong data sub-directory (this routine can be used on the normal QL, too). One hard disk partition and both floppy drives are accessible to Qdos.

Readers' letters

Vladimir Sams reports from Belgrade that he has now got his 'home' QL running, after what appeared to be display-induced failure. The MC1377P chip (IC28) was replaced by a local repairer. He mentioned that the Washington DC QL group in the USA has details of a buffered display cable, to prevent displays killing chips through the unbuffered QL interface. Sams now has another problem – how to get two battery-backed clocks (bought in the USA) working. Does anyone know of 'standard' problems with such clocks?

P Land has got his printer spacing correctly (it previously double-spaced with Quill and Abacus but not with SuperBasic programs). His problem was quite common, but no easier to solve, at first, just because it was a simple one. It was a question of what code(s) the printer was set to expect, to move one line space. Typical dot-matrix printers have an internal switch setting which allows them to Line Feed, or not to Line Feed, when a Carriage Return command is received from a computer. If the printer is set to add a Line Feed to every Carriage Return automatically, and the computer program is set to do the same, the result is double-spacing instead of single. You can set the Quill printer-driver to give an EOL (end of line) command to the printer of CR + LF, or just CR. To match just a CR, the printer itself has to respond to the pressing of the ENTER key with a Carriage Return (the print head returns to its leftmost position) and a single Line Feed. Land set the Quill printer-driver to give only CR when the ENTER key is pressed; this is an easier solution than getting into the bowels of the printer to reset a DIP switch, but fixing the problem this way might result in trouble printing from other programs in future, and there is no guarantee that the other programmers will have provided the facility to set the EOL code(s).

Since writing comments about using fax paper rolls for the Serial 8056 printer in a previous issue, a fax machine has been added to my own collection, making me more familiar with the subject. The Amstrad FX 9600AT can accommodate both 210- and 216 mm -wide rolls, with the aid of two spacing pieces supplied as standard with it. The thermal paper rolls are 30 m in length and cost £3.60 + VAT (210 mm) from my local computer shop. The more usual retail price appears to be about £5–6 per roll. A 30 m roll is 5.5 cm in diameter.

Perhaps equally important is the internal diameter of the core of the roll, which is 2.5 cm in the case of the Amstrad fax machine. Letters received so far seem to imply that only roll width matters, but a supplier of fax paper (see Information box below) has been kind enough to write in response to the note about **R Thompson,** and the letter suggests that ½ in is the required core diameter for the 8056. If you are having trouble getting paper for your 8056, and decide to contact this supplier, it might save time to inform them of the internal core diameter of the 'proper' rolls, and supply a sample piece of paper.

As an aside to this matter, the Amstrad machine has a din connector on the back 'for passing scanned images to the computer'. This was news to me, and I suspect that there are no suitable interfaces available for processing the fax-machine output so that it could be accepted by a graphics program. Does anyone have experience of feeding images scanned by a fax machine into a computer?

Mike Jackman had not (as of late April) managed to find a solution to his problem with *TechniKit/TechniQL*, from **Talent Computer Systems.** He successfully uses the Plot module of the program on his QL to produce output from a Penman plotter, but cannot get QL screens dumped from the Screen Dump module. The program 'does not run at all' on his Thor. Can anyone shed any light on this subject?

The only well-known QL supplier left that supplies a wide range of software written by people who have since left the QL scene is **TK Computerware.** It is good to have a supplier willing to deal in obsolete items, but it is almost inevitable that the supplier will come in for criticism from customers who feel they are not getting the usual speedy service, when they ask for help with old programs. When a program is actually produced by a supplier, the customer has good reason to hop up and down if complaints are not dealt with quickly, but there is nothing much a supplier can do when it cannot get a good response to enquiries about old programs, from other suppliers or programmers who may no longer be interested in QL matters.

Some users seem to develop a love-hate relationship with all or part of their Computer system, and **P H Tanner** seems to fall into this category where Minerva is concerned. He emphasises that he thinks that the product is '24 carat', but is far less complimentary about the support for it. In his running battle to get his two QLs talking to each other and doing (Forth-language) work he wants to get on with, he has at least achieved one success, in contradiction of 'what the experts have told him'. It has long been my understanding that a decent network between two QLs is not possible unless both of them are fitted with *Toolkit II* , on rom. Tanner says he now has a good network, without TKII on rom being in both computers. He doesn't say whether he is using TKII at all, however, and it may be worth pointing out for the benefit of other would-be networkers that the basic QL network commands do work, after a fashion, without any assistance from any toolkit. The troubles are that: the network set up in this way is not reliable; the available commands are very limited; and the QL-network hardware varies in its reliability from computer to computer. Another step in the right direction for him is that he has found a way of getting *QFlash* to work with Minerva.

Italy has a fair number of QL enthusiasts, one of whom is **Francesco Querini,** who has written with some comments on possible future hardware developments. He has designed various units for **SPEM,** an Italian company which has been producing hardware for the QL for several years now. One idea he mentioned is an advanced graphics driver board. Undoubtedly, many users would like to have a sharper screen picture, and that is now possible on the Atari ST with QL emulator, but not on the QL itself. With the greater speed and memory now available from the *Gold Card* , it would be nice to do more justice to programs such as Professional Publisher, but would the types of display most of us have (eg Philips CM8833, Microvitec Cub 653) be able to give a sharper picture?

INFORMATION
Fax paper for Serial 8056: ACD Morgan, Sensitised Coatings Ltd, Bergen Way, North Lynn Industrial Estate, King's Lynn, Norfolk PE30 2JL. Tel: (0553) 760377.

QL SCENE

Two new lines from star printers

Star Micronics UK has announced two new printer lines. The LC-20 is dubbed 'the CL-10 for the 90s'. A new 9-pin entry-level dot matrix printer in Star's Business Series, the LC-20 is aimed at small businesses, home users, education and small departments in large companies. It has eight near-letter-quality fonts including italics, print speeds of 180 cps in draft and 45 cps in nlq (12 characters per inch), a 4 KB print buffer, paper parking, push tractor feed with short form tear-off, a built in parallel centronics interface and Epson and IBM emulation. An optional serial interface is available.

The LC-20 has a robust case and quiet mode printing, and is priced £199 exclusive of VAT. The LC-200, launched in September 1990, has the same specifications plus colour printing, for £259 ex. VAT.

Star have also launched a range of four 9- and 24-pin printers with colour facilities as standard. Priced between £399 and £599, the Pro-to-Col series replaces the Professional Series, and includes options such as buffer expansion to 236 KB, high speed draft mode, up to 14 fonts, 80 column model and wide carriage model. The Pro-to-Col is designed for 'general high quality office print needs'.

For more information, contact **Star Micronics UK, Star House, Peregrine Business Park, Gomm Road, High Wycombe, Bucks HP13 7DL. Tel. 0494 471111.**

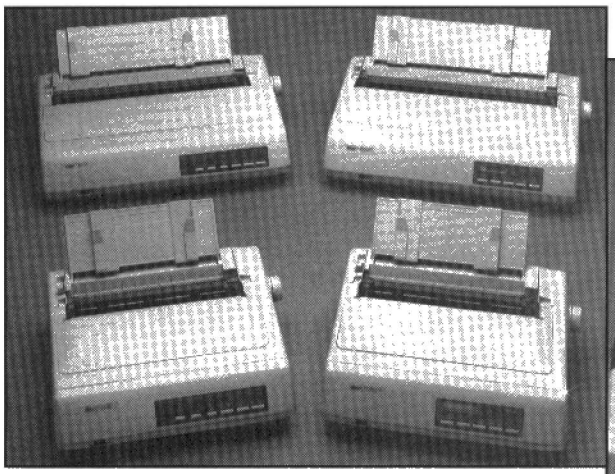
Price changes at CGH

CGH Services have issued a new price list from July 1991. The main points of interest are that they are no longer supporting 'base' prices, where purchasers send their own media. This applies to the public domain library as well as commercial sales. The public domain library is no longer supporting mdvs, and will charge £2 each (including UK postage and packing) for disks supplied. 'We'll pack them as tight as we can,' stresses Richard Alexander.

All commercial prices will now have UK postage included. There is no VAT, as CGH are below the VAT threshold. Non-UK customers add 10% (Europe) or 20% (elsewhere) to cover postage.

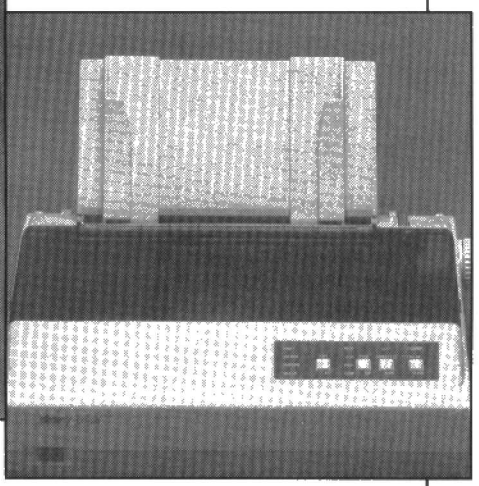
CGH are also revising their authors' royalties structure to remove anomalies.

For a price list contact **CGH Services, Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA. Tel. 0559 384574.** Send an SAE or international reply coupons for their full public domain catalogue.



Above: THE STAR PRO-TO-COL SERIES

Below: THE LC-20



Cowo release new QTop version

Cowo Electronic have produced the first major revision of their user front-end package *QTop*, the more powerful VI.10. The full QTop system is based on six stand-alone executable multitasking programs. QTop-Desk is the system manager, offering main menus for Desk, Devices, Files, Jobs, Keyboard, Programs, Myprogs, Tools and Options, with over 110 submenus. Cowo believe that the Files and Jobs handling routines are 'the most powerful and complex in the whole QL world.' Real sub directories on all QJump Level 2 device drivers (Miracle Hard Disk, *Gold Card*, Atari ST/QL emulator) and on Thor hard disks are supported.

QTop-Clocks is a multi-functional clock program offering four different digital clocks, an analogue clock and an alarm clock. QTop-Demos offers ten different high-speed graphics demonstrations. QTop-Animator is a flexible graphics animation program. QTop-index tests the hardware of your computer system. QTop-Snap is a screen dump and save program. The latest QJump Pointer Environment is included — it is not necessary to run the Pointer Environment, it is fully supported if installed.

QTop is fully user-configurable, and includes a screen refresh and freely moveable and resizeable windows.

The software with a 41-page manual is available in English or German for £35 (on 3.5 or 5.25 in disk) or £39 (on three microcassettes), prices inclusive of post and packing.

For a six-page information brochure and order form, contact **Cowo Electronic, Urs Konig, Munsterstrasse 4, CH-6210 Sursee, Switzerland.**

OPEN CHANNEL

Open Channel is where you have the opportunity to voice your opinions in *Sinclair QL World*. Whether you want to ask for help with a technical problem, provide somebody with the answer, or just sound off about something which bothers you, write to: Open Channel, Sinclair QL World, 116/120 Goswell Road, London EC1V 7QD.

Thank you

As a fairly average QL user, most of my purchases over the years have been promoted by articles and reviews in your pages. As a result of Mark Knight's May 1991 review, I recently bought *Perfection* and have found it to be the definitive answer to all *Quill's* shortcomings, *Pro Publisher*, with perseverance, can be splendid in its own special field, but Perfection (unlike other attempts to replace Quill) is child's play to the relatively inexperienced wp user, yet it is still capable of great sophistication. It is, quite simply, marvellous, especially when coupled with *Lightning*. Thank you for Perfection, and thank you, too, for telling me about Pro Publisher, the elegantly simple multi-tasking Taskforce, and my splendid (and cheap) LC24-10 printer.

Brendan Connors
Hove
Sussex

Editor's comment: That's why we run full-length user-driven printer reports every so often.

Listing

Can anyone help me with a program for listing my SuperBasic programs? I have a C-Itok daisywheel printer and have tried the following program, which does in fact provide a carriage return after each line.

Baud 9600: Open #3, serl: list #3: close #3

Result:

```
10 Rem prog list
 20 next line
  30 next line
   40 next line
50 next line
```

etc.

This listing actually runs off the page and is lost, leaving me very frustrated. The printer works normally with *Quill* and *Abacus*. Any suggestions from readers would be appreciated.

Malcolm Roberts
Pembroke
Dyfed

Thank you 2

I should like to express my gratitude through your column for the excellent service and expert help given to me by Mr. Freddie Vachha of Digital Precision Ltd. I had been using a single 3.5 in disk drive with a 768K Trump Card on my QL and had ordered from them copies of *QFlick* and *Perfection*. On reciept, neither would load, and the error message 'not found' was given.

On the telephone, Mr. Vaccha advised (a) cleaning and (b) realigning the reading heads. Cleaning proved ineffective, but I had on order from Miracle Systems a dual 3.5 in drive unit, and when this arrived and was connected, *Perfection* loaded and operated perfectly.

In addition to his 'remote diagnosis' of my problem, Mr. Vaccha suggested a firm who would re-align my single drive unit, and I feel his expertise is largely instrumental in keeping alive an excellent computer.

I also greatly appreciate your series *The New User Guide* for people who only have elementary programming skills, and would like to become more proficient.

I receive *QL World* by annual subscription, and have completed the QLAW questionnaire.

R A Mills
Heighington
Lincoln

Positive

The very positive review of *Perfection*, Digital Precision's new word processor, in these pages encouraged me to order what appeared to be a very worthwhile package.

I have now received it and find it excellent, easy to use and the answer to my wordprocessing problems.

It is a pleasure to find software whose performance exceeds even the suppliers' advertising blurb.

J R Haldane
Gorebridge
Midlothian

Interest

This concerns the reader's problem in the August issue under the heading 'Capital'. I have two programs which I contributed myself, which I find very useful, and I hope Mr. Patterson will find them useful too. One is for *Abacus*, while the other is a simple SuperBasic program.

The Abacus program gives accrued interest for a year of 12 equal months, which results in

Editor's notebook

There are ways you can tell that the recession is biting. House-owning colleagues speculate on the possibility of selling up and moving to somewhere rented. Holidaying colleagues are under canvas in Wales instead of under blue skies in Greece. One well-off acquaintance returned from Greece moaning that he wished he'd stayed there. Unusual? You bet. Normally you can't lever him away from his whizz-o-phone.

He works for a big – I mean *very* big – computer company. Their business isn't what it was. Oddly, although we hear the occasional lament that things are slow, QL business seems to be moving along steadily.

Normally the long, hot days of August are the cue for silence from the outside world. Not so this year – QLers are sitting at home writing! So we have the first of a new series of machine code programming, a handful of reviews and a monster DBQL, as well as the regulars.

The Formulae used

	A	B	C	D	E	F	G	H	I	J	K	L
1	1991	*	Month	Capital		rate%		accrued		In/out(-)	*	Growing
2		*	==========								*	Accrued
3		*	B/f			£0.00					*	==========
4		*	Jan	D3+H4+J4		0.00		(D3+J4)*F4/1200			*	H4
5		*	Feb	D3+sum(H4:H5)+sum(J4:J5)		F4		(D3+J4+J5)*F5/1200			*	L4+H5
6		*	Mar	D3+sum(H4:H6)+sum(J4:J6)		F5		(D3+sum(J4:J6))*F6/1200			*	L5+H6
7		*	Apr	D3+sum(H4:H7)+sum(J4:J7)		F6		(D3+sum(J4:J7))*F7/1200			*	L6+H7
8		*	May	D3+sum(H4:H8)+sum(J4:J8)		F7		(D3+sum(J4:J8))*F8/1200			*	L7+H8
9		*	Jun	D3+sum(H4:H9)+sum(J4:J9)		F8		(D3+sum(J4:J9))*F9/1200			*	L8+H9
10		*	Jul	D3+sum(H4:H10)+sum(J4:J10)		F9		(D3+sum(J4:J10))*F10/1200			*	L9+H10
11		*	Aug	D3+sum(H4:H11)+sum(J4:J11)		F10		(D3+sum(J4:J11))*F11/1200			*	L10+H11
12		*	Sep	D3+sum(H4:H12)+sum(J4:J12)		F11		(D3+sum(J4:J12))*F12/1200			*	L11+H12
13		*	Oct	D3+sum(H4:H13)+sum(J4:J13)		F12		(D3+sum(J4:J13))*F13/1200			*	L12+H13
14		*	Nov	D3+sum(H4:H14)+sum(J4:J14)		F13		(D3+sum(J4:J14))*F14/1200			*	L13+H14
15		*	Dec	D3+sum(H4:H15)+sum(J4:J15)		F14		(D3+sum(J4:J15))*F15/1200			*	L14+H15
16		##########										

A building society account on Abacus

Some Figures

	A	B	C	D	E	F	G	H	I	J	K	L
1	1991	*	Month	Capital		rate%		accrued		In/out(-)	*	Growing
2		*	==========								*	Accrued
3		*	B/f	£21008.32							*	==========
4		*	Jan	£21175.69		9.56		167.37			*	167.37
5		*	Feb	£22351.02		9.56		175.33		1000	*	342.70
6		*	Mar	£22509.11		8.62		158.09			*	500.79
7		*	Apr	£22667.21		8.62		158.09			*	658.89
8		*	May	£22246.62		7.80		139.41		-560	*	798.30
9		*	Jun	£22386.03		7.80		139.41			*	937.71
10		*	Jul	£23531.95		7.80		145.91		1000	*	1083.63
11		*	Aug	£23677.86		7.80		145.91			*	1229.54
12		*	Sep	£23823.78		7.80		145.91			*	1375.46
13		*	Oct	£23969.69		7.80		145.91			*	1521.37
14		*	Nov	£24115.60		7.80		145.91			*	1667.28
15		*	Dec	£24261.52		7.80		145.91			*	1813.20
16		##########										

a 360-day year. This will be close enough for most purposes. The interest rate will have to be amended each time it changes and each time that happens the program automatically amends the rate for the remaining months of the year, because each cell in the column uses the reference of the one above it. Thus, one has a forecast of one's position for the year-end.

The accrued interest figures are meant to be taken as at the close of business on the last day of the month in question; the last day of December is therefore the last day of the year.

Deposits and withdrawals are entered in the 'in/out' column, but in order not to make the program unnecessarily complicated, no provision has been made in the column to stipulate on which day of the month the transaction was made.

Each row can be typed one at a time, or one can go as far as March and then COPY the row for March to get April, editing month-name and cell references where necessary, ensuring that D3 and J4 are referred to in every line thereafter. Likewise, one can COPY March and April to get June and July - and March, April, May, June and July to get August, September, October, November and December.

Once I was sure that I had the programs I wanted, I made a COPY of the whole thing twice on rows below the ones in use, and made the figures in them zero. This is a useful procedure, because one cannot LOAD a second file on to the spreadsheet, and keep the one on it which is already in use, as far as I know. With the procedure one can always COPY a blank for next year onto the rows below.

The SuperBasic program will be self-explanatory once it is run.

Peter Tomlin
Hatherley Grove
London

A compound interest table in SuperBasic

```
100 MODE 4: INK 4: PAPER 0: CLS
110 INPUT "Sum", x
120 INPUT "Years", y
130 INPUT "Rate", t
140 LET r=t/100
150 CLS
160 FOR i=1 TO y
170 LET s=x*((1+r) ^i)
180 PRINT "  Year"!i,, "£"; s
190 END FOR i
200 PRINT
```

Strings

I have found a Bug in Superbasic I have not seen detailed before. It concerns the comparison of strings. Some characters are considered equal by the interpreter, which are obviously not I stumbled on this while writing a guarded input routine; the character O is effectively the same as the cursor up key. The only way to distinguish is to compare CODEs. I have since found a total of 64 cases (see enclosed printout). There is an interesting effect: for example, A is equal to CHR# 225, and a equal also to CHR# 225 but A is not equal to a (using == completes this).

A few months ago Mike Lloyd gave a function for signum: REturn 1-(x<=0)-(x<0). This does have an elegant simplicity, however, the alternative, RET (x>0)-(x<0) works just as well.

Caleb Leeke
Huntingdon
Cambs

```
. CHR$(46)  = CHR$(206)
0 CHR$(48)  = CHR$(208)
1 CHR$(49)  = CHR$(209)
2 CHR$(50)  = CHR$(210)
3 CHR$(51)  = CHR$(211)
4 CHR$(52)  = CHR$(212)
5 CHR$(53)  = CHR$(213)
6 CHR$(54)  = CHR$(214)
7 CHR$(55)  = CHR$(215)
8 CHR$(56)  = CHR$(216)
9 CHR$(57)  = CHR$(217)
A CHR$(65)  = CHR$(225)
B CHR$(66)  = CHR$(226)
C CHR$(67)  = CHR$(227)
D CHR$(68)  = CHR$(228)
E CHR$(69)  = CHR$(229)
F CHR$(70)  = CHR$(230)
G CHR$(71)  = CHR$(231)
H CHR$(72)  = CHR$(232)
I CHR$(73)  = CHR$(233)
J CHR$(74)  = CHR$(234)
K CHR$(75)  = CHR$(235)
L CHR$(76)  = CHR$(236)
M CHR$(77)  = CHR$(237)
N CHR$(78)  = CHR$(238)
O CHR$(79)  = CHR$(239)
P CHR$(80)  = CHR$(240)
Q CHR$(81)  = CHR$(241)
R CHR$(82)  = CHR$(242)
S CHR$(83)  = CHR$(243)
T CHR$(84)  = CHR$(244)
U CHR$(85)  = CHR$(245)
V CHR$(86)  = CHR$(246)
W CHR$(87)  = CHR$(247)
X CHR$(88)  = CHR$(248)
Y CHR$(89)  = CHR$(249)
Z CHR$(90)  = CHR$(250)
_ CHR$(95)  = CHR$(255)
a CHR$(97)  = CHR$(225)
b CHR$(98)  = CHR$(226)
c CHR$(99)  = CHR$(227)
d CHR$(100) = CHR$(228)
e CHR$(101) = CHR$(229)
f CHR$(102) = CHR$(230)
g CHR$(103) = CHR$(231)
h CHR$(104) = CHR$(232)
i CHR$(105) = CHR$(233)
j CHR$(106) = CHR$(234)
k CHR$(107) = CHR$(235)
l CHR$(108) = CHR$(236)
m CHR$(109) = CHR$(237)
n CHR$(110) = CHR$(238)
o CHR$(111) = CHR$(239)
p CHR$(112) = CHR$(240)
q CHR$(113) = CHR$(241)
r CHR$(114) = CHR$(242)
s CHR$(115) = CHR$(243)
t CHR$(116) = CHR$(244)
u CHR$(117) = CHR$(245)
v CHR$(118) = CHR$(246)
w CHR$(119) = CHR$(247)
x CHR$(120) = CHR$(248)
y CHR$(121) = CHR$(249)
z CHR$(122) = CHR$(250)
```

THE GOPHER

INFORMATION
Program: *The Gopher* V1.02
Price: £12 (add £1 postage for EEC countries,£2.50 elsewhere).
Supplier: Dilwyn Jones Computing, 41 Bro Emrys, Tal-y-Bont, Bangor, Gwynedd LL57 3YT. Tel: (0248) - 354023.

Bryan Davies investigates a species that tracks down elusive disk files and reports them back to base.

The hard disk user is in a paradoxical situation. It is perhaps the most expensive add-on unit for the QL, but there is little software available for it, to spend more money on. At the time of writing, Dilwyn Jones Computing offer the only two items that users are likely to get to hear about. *WinBack* was dealt with in a previous review; it is a fairly basic hard disk backup program, which works well. *The Gopher* was written by the same person, so one would expect it to be of a similar nature and, indeed, the similarity is clear.

The Gopher is available on microdrive, and SuperBasic and compiled with Digital Precision's *Turbo*. It requires the usual Turbo RUNTIME_EXTS file to be loaded, plus an additional one called GOPHER_BIN, for SB commands specific to The Gopher. If a suitable version of the XTRAS or TURBO_TK_CODE file has already been loaded, the runtime file is not needed; the boot routine asks you whether or not it should go ahead and load it. About 32 KB of memory is required to run the program. In case the origin of the name is not yet clear, the program's function is to 'go for' files, that is, ferret out files which the user cannot (easily) locate. While such a utility obviously appeals most to the hard disk user, because of the much greater number of files this drive holds than either floppy or microdrive, there are some users who manage to get a hundred or so files on their floppies and may have difficulty locating required ones.

Booklet

There is an 11-page instruction booklet, which covers program operations adequately. An UPDATES_DOC file is provided on disk (or cartridge) to bring the purchaser up to date on happenings since the manual was printed. The style is informal, without being chatty, and lets the purchaser in on some of the program development details. The manual includes considerable explanation on using the clone program supplied, but it is easy simply to COPY OR WCOPY the files to a working disk or cartridge. The manual doesn't deal with the purpose of two files on the supplied disk, and you have to read one of those files to find out what the other is for. It is a program for altering the data space taken by the executable files; users of the Turbo compiler will find this quite understandable, but average *Quill* users may find it a complete mystery. There is no mention of why users of The Gopher should need to change data space, so the best thing the user can do is forget it.

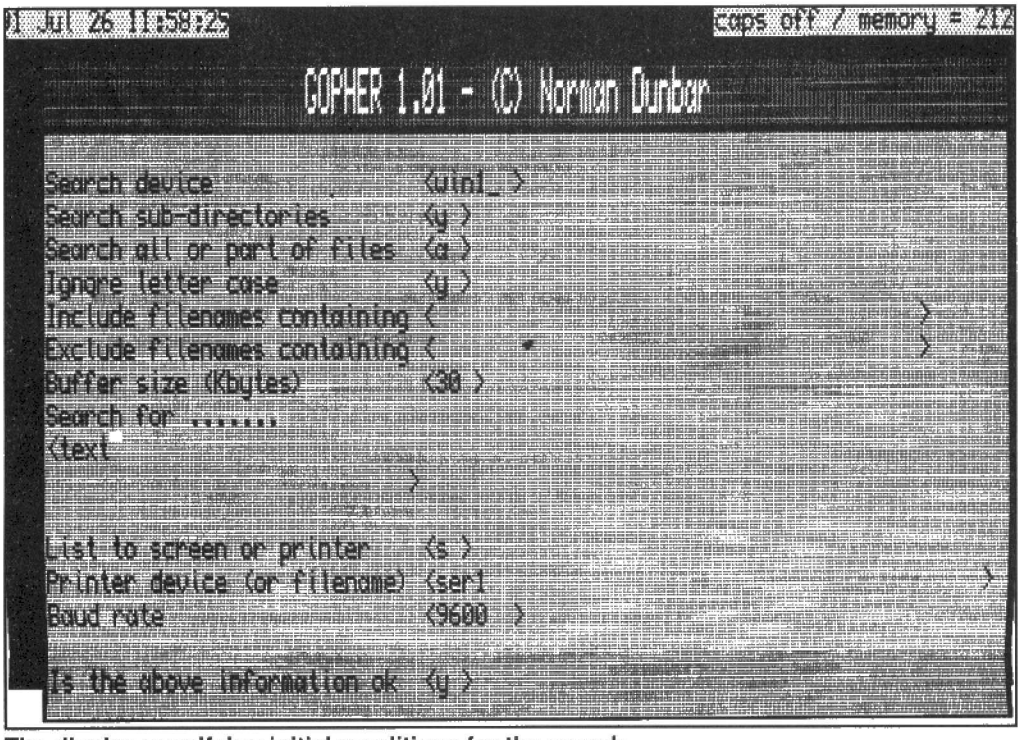
If finding a file were only a matter of tracing a particular file name, it might be indulgent to employ a program to do the job for you. A reasonably organised user will give files names which are not hard to relate to file contents. The flysheet on The Gopher supplied by DJC gives the example of tracing letters written to *QL World* (a fair task, in my case). The restriction of eight characters for a file name means that you can't hope to provide much, if anything, of a clue to the file contents in its name; at best, you can have something like QLWLET1.DOC etc. You'll know whether or not a file is indeed a letter to QL World, but won't have a clue as to a letter's contents. The program must, therefore, be capable of looking *into* files, and checking there for key text strings, from which the file contents can be gauged. It is not presently usual for QL word-processing programs to provide a 'comments' facility, separate from the file text itself, and The Gopher has to go through the text until it finds, or fails to find, any specified character string. With long files, this could be a lengthy process.

To speed-up the search process, it is desirable to exclude any groups of files which are unlikely to contain what is being looked for. Maybe you always write your letters to QL World with *text* [87], in which case you can specify that only files with the _T87 extension in their names are examined. In case the impression given so far is that the program is a tool for word-processing users only, rest assured that it can be used with any files, but string searches are obviously limited to what characters can be entered into the string. The manual doesn't list the acceptable characters, but 'specials' such as 0 can be used. The search string can be up to 100 characters in length. The search used when producing the screen dumps given with this article took about six minutes to complete. There were 192 files on the hard disk, occupying 2.7 MB, and 35 instances of the search string were reported as found. A more specific, case-dependent search, for the string INDEXICAL, took well under one minute, when the file suffix (_WP) of the only file found had been specified, whereas it took seven minutes with no restrictions made.

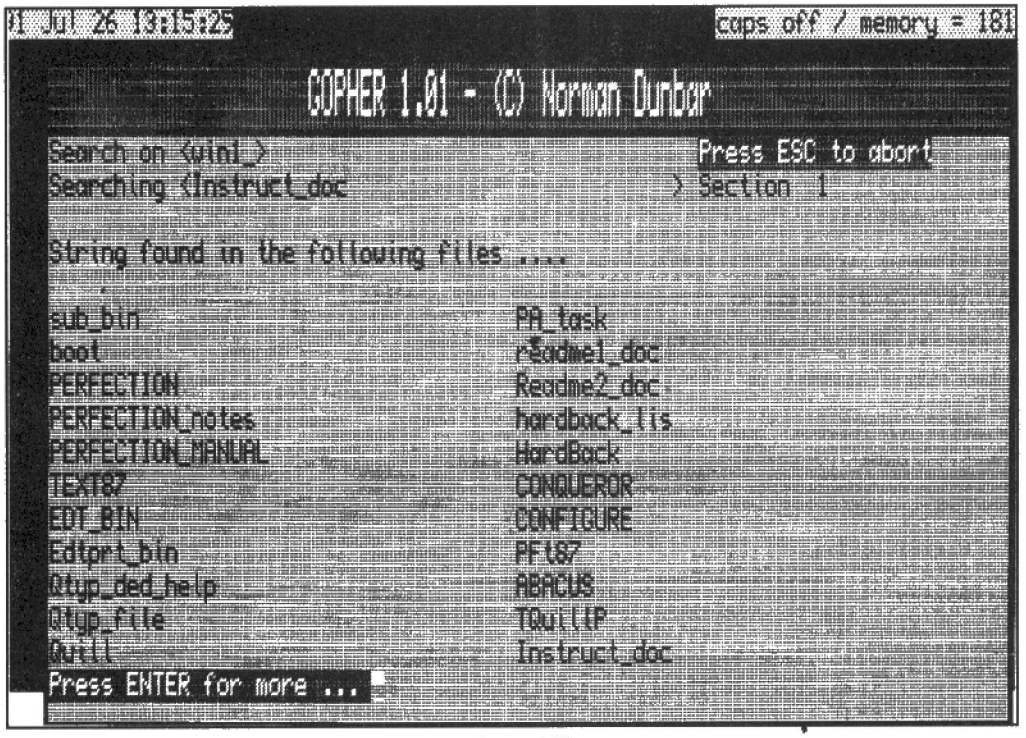
Figure one is of the display at the stage of specifying initial conditions for a search. The range of conditions is greater than one might expect. For in-

stance, you can speed-up a search, if you are interested in finding the search string only in the first few pages of documents, by opting to search only the first part of each file, and specifying a buffer size sufficient just for the pages you want searched. The default buffer size for a part-search is 2 KB. The minimum is 1 KB, nominally equivalent to about 150 words of text (but bear in mind that text files tend to have formatting information at the front and even a file with no text in it may occupy 2Kb).

The buffer action is different if whole files are to be searched; the search is then fastest if the buffer is set to the maximum 30Kb. The program loads files in chunks of the buffer size specified. When one chunk has been examined, the next is loaded. The decision on case-sensitive or not affects search speed, too. Entering <N> here – that is, case is to be taken into account – is stated to reduced the search time by as much as a half.



The display specifying initial conditions for the search.



The search has found the string in a number of files.

Legal

The 'legal' entries for Search Device are any valid QL device with a five-character name; the underline must be the last character, and the drive number must be specified (in the range) 1-8). You cannot enter a sub directory name, therefore. Those users who have successfully constructed an extensive sub directory structure will be glad the option is provided to specify whether or not (all) sub directories are to be searched. It is pointed out in the instructions that you can check all sub directories of a Miracle Systems hard disk by using CTRL-C to switch to SB, then typing <WIN2> and pressing ENTER, then returning to the program. The point here is that the hard disk software may have been set for a particular sub directory, before The Gopher was started, and the program will then not be able to look at the whole disk. At present, the sub directories option is available only if you have entered <WINn> as the Search Device. Presumably, provision will be made before long for the high-density floppy disk drives with the Miracle Systems *Gold Card*, since they have the same sub directory structure available as for the hard disk.

Apart from entering the search string, all you need do much of the time is press ENTER for all entries. When the search string is located, the name of the file it was found in is written to the screen, or to both screen and printer, as specified. If the printer option is selected, you can opt to send the files names shown on-screen to either printer or disk drive file. If there are enough instances of the string being found, you are requested to press ENTER to clear the screen and make way for any further instances. The maximum number of files displayable per screen is 22. **Figure two** shows the display when a search has found more than the 22 instances of the search string noted in the first illustration – <text>. In this case, whole files were searched, with the buffer set to 30 KB. **Figure three** shows how setting the program to do part-searches, with a buffer of 2 KB, affected the number of instances found.

Unlike many small programs, The Gopher does have a refresh key (the standard F4) to clean the screen if another program has corrupted it. Being a multi-tasking program, it is quite likely The Gopher will have company in the QL. In fact, the review was done with

my normal system fully loaded, and this didn't seem to upset The Gopher at all. As the *Perfection* WP program was being reviewed at the same time, it was loaded, as was its accessory printer-driver creation program, *Flashback 2*, *Files 2*, *Q-Switch*, and a *lot* of extensions. Apart from Files 2 disappearing on one occasion (possibly not connected with The Gopher), there was no apparent interaction between programs. Error trapping is incorporated in the program, so you are unlikely to be caught out by things like failing to put a disk in a search drive.

Killed

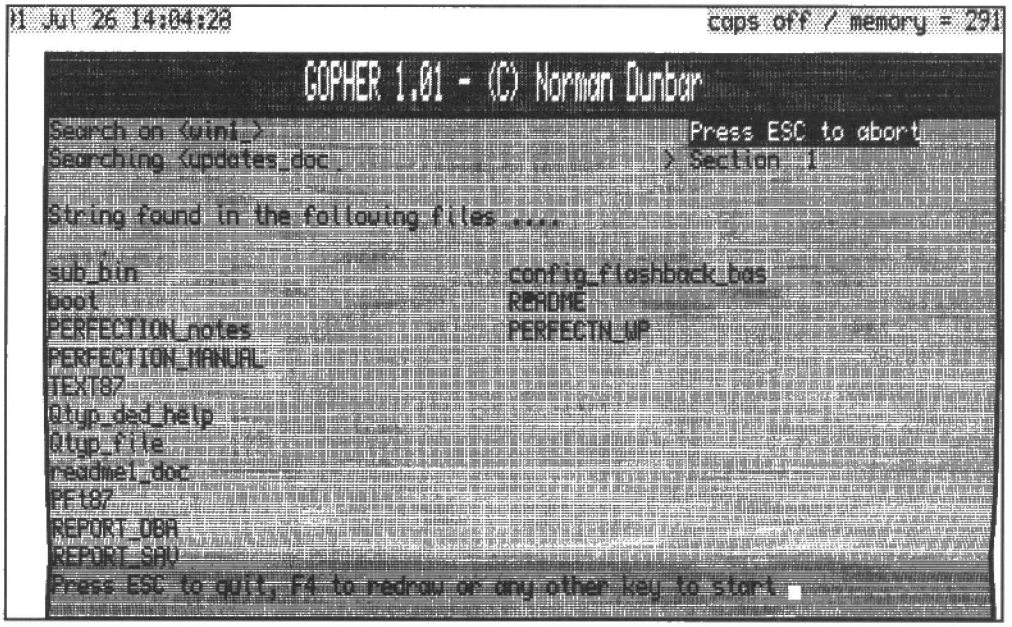
The Gopher clearly did not like one file, namely QTYP _DICTIONARY, which is to be found as part of QJump's *Qtyp* spelling checker and later versions of text[87]. The program stopped when it came across this file on hard disk, and gave the message 'file error'. The option was offered to press ENTER in the hope the search could be continued, but the attempt to resume was unsuccessful and the program had to be killed and restarted. This happened with subsequent attempts too, making the program unusable until the QType dictionary had been deleted from the hard disk. Rather an extreme reaction. However, an earlier version of the file on a floppy gave no trouble and copying that to the hard disk removed the problem. Maybe the file was corrupted in some way; it had been edited, but with the QType dictionary editor.

All files are OPENed to be checked. Users of Archive who have lost files through not having them CLOSED successfully might be made nervous by the thought of so many files being opened, but there may be no alternative to doing this. On the matter of what characters can be entered as part of a string, some users might wish for non Ascii characters such as the alphabetic capitals with a bar over the top, as used by *The Editor*.

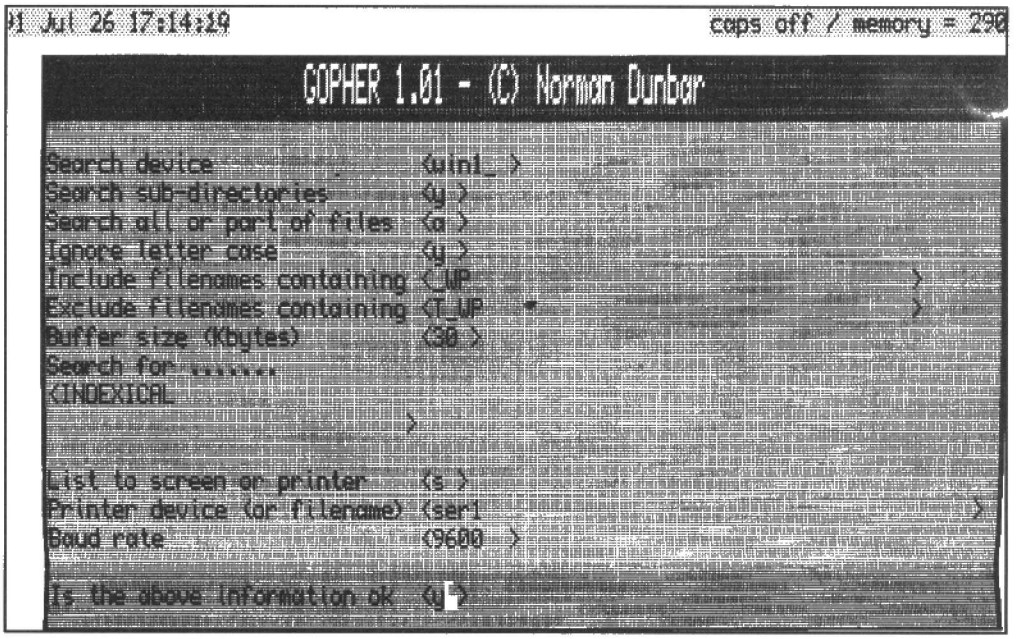
Neither DJC nor the writer of the program go over the top in making claims for it, nor do they ask a lot of money for it. There are some features one would like to see incorporated into it, but it would be only reasonable to expect the price to go up if they were added. One obvious wish is for a display of the location of a string, when it is found. No doubt adding such a feature would be time-consuming, both in the programming required and in the effect it would have on the speed of the program, but it might save having to load into several files which were found to have the desired string in them, in order to discover which instance is the required one.

A smaller wish is that the search string should be displayed during the search. There are some of us who tend to forget what we have done, very quickly; it is also desirable to be able to check whether or not one put spaces (for instance) in a string. The instructions could have been clearer on the matter of how Include and Exclude entries can be typed in; there was space for 36 characters, which is the Qdos file name limit but is more than would be used in a normal file name (12 maximum). Presumably, only single, complete strings can be used. It would be handy to be able to enter several strings – eg _DOC _EXT _T87 _BIN – and have them all, individually, excluded/included. **Figure four** shows how these entries are actually used. All files having <T_WP> as part of their names would not be checked, but *any remaining* files having <_WP> would be checked. This excluded *all* files which did not have <_WP> in their names, and also excluded any files which had <T_WP>.

The Gopher is a fairly basic program, without pretensions to greatness, but it is straightforward to use, and works well. There is the possibility of additional features being added in upgrades; presumably this will depend upon how well the program sells in its current version. At the price, it is good value.



The results of a part-search with a 2 KB buffer.



The display showing the include/exclude options.

QL SCENE

Solent sub-group launches new Quanta workshop

The Solent sub-group of Quanta is holding its first Workshop on Sunday 22 September from 10am to 6pm at the Horizon Centre, Sundridge Close, Portsmouth. The modern venue has 'excellent facilities', including facilities for the disabled, and plenty of car parking. It is close to the A3 and the M27 and is also handy for cross-Channel ferries from Europe. Food and drink will be available all day, and the licensed bar will be open at lunch time.

There will be a number of lectures or presentations on a wide range of QL-related topics by the well-known and not so well-known. Traders will be present and there will be new products to see and assess. There will be facilities to get together and discuss recent new developments in the QL market.

The organisers also emphasise that these events are called 'workshops' for a good reason – they encourage attendees to bring their QLs and relations along, 'set them up and have a good time'. There will be a Clinic Desk for problem-solving, and competitions and games for youngsters, who are also welcome.

For information about the event, transport and accommodation needs, etc., please contact **Graham Goodwin 0489 895451, Graham Evans 042121 3350 or (regional transport, accommodation and other local events) Portsmouth Tourist Information office 0705 838382.**

Other workshops

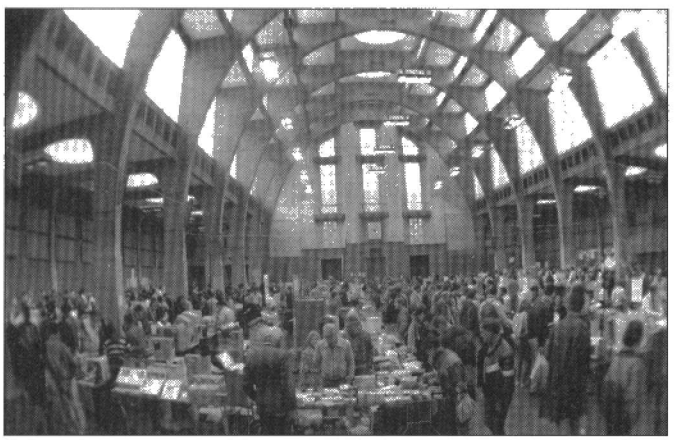
Other Quanta workshops planned for the near future are **Birmingham** (28 September) at the Sandwell District General Hospital Postgraduate Centre (call Bill Fuggle 021 4726671 or Dennis Briggs 0952 255895 for information), **Nottingham** 12-13 October (no further information given) and **Bristol** 17 November (no further information given).

Fairing further

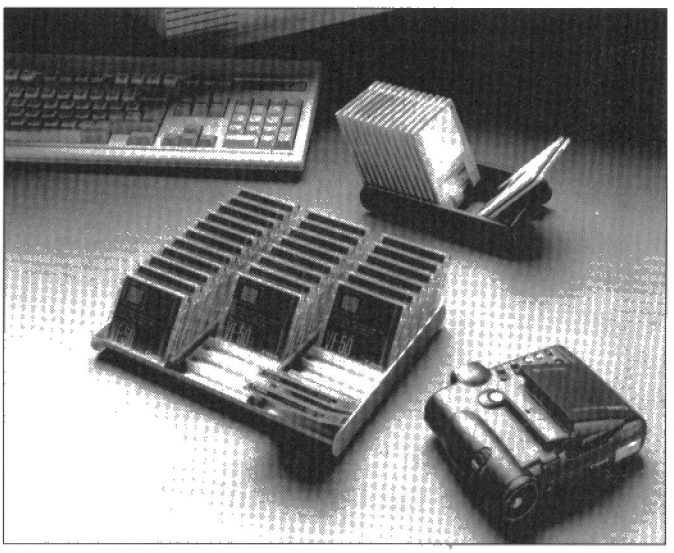
The All Formats Computer Fair has extended its range again. After two successful fairs in the Midlands, the All Formats Fair is now happening at five venues around the UK. At least four visits to each venue a year are planned. The next few fairs are:

Midlands: National Motorcycle Museum, Solihull, M42 (opposite the National Exhibition Centre and Birmingham International BR station), Saturday 14 September; **Scotland**: City Hall, Candleriggs, Glasgow, Sunday 22 September; **Western**: The Brunel Centre, Bristol Old Station (near Temple Meads Station), Sunday 6 October.

The admission charge is now £4 but the stand price has dropped to £60. All fairs start at 10am and finish at 4pm. All venues have ample car parking. For advance tickets and stands, contact **John Riding on 0225 868100.**



Little boxes



Lift UK Ltd, established makers of storage systems for music and video recording, are launching a range of storage units for disks.

The Databoy range includes desk units for 2 in, 3.5 in and 5.25 in disks. The ridged construction means that disks can be flipped forward for browsing, without the row collapsing.

23-disk units will cost £3.99 and 43-disk units will cost £5.99. Models with protective lids are expected to follow. Lift are also advertising a logo-stamping service for promotional purposes.

Information from **Lift (UK) Ltd, Triangle Business park, Wendover Road, Stoke Mandeville, Bucks HP22 5BL. Tel. 0296 615151.**

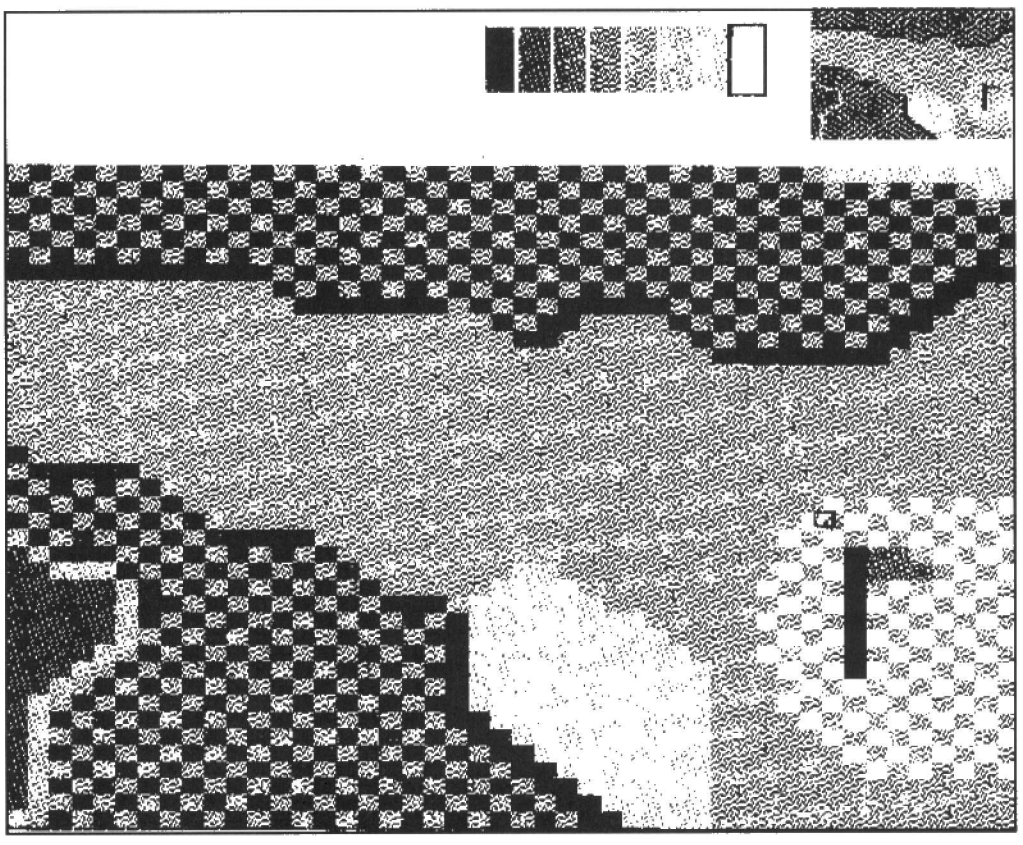
SOFTWARE FILE

THE PAINTER

INFORMATION

Program: *The Painter* V4.01, for QL with 256K+ expansion.
Price: £45 (+£10 bank costs) or 3000 Belgian Francs. Add 300 BFr if outside EEC. Cheques payable to Van der Auwera.
Publisher: Progs, Haachtstraat 92, B - 3020 Veltem, Belgium. Tel: (016) 48 89 52.

Rich Mellor has been attracted to the latest version of The Painter by its upgrades.



The Painter: Edit pixels one by one in Enlarge mode and see the effect immediately.

Throughout the QL's long life, there have been several drawing packages which varied a great deal in utility, speed and number of features. *The Painter* is the most recent in a long line up which includes such illustrious names as *GraphiQL Plus, MiceArt, M-Paint* and *Eye-Q*.

In 1989 I went to the Stafford Computer Show and saw The Painter. I was hooked after a few minutes using it, but decided not to purchase it at the time. Since then, several improvements have been made and when I received a special offer to buy a copy recently, I decided to take it up.

The current version is now fully *Minerva* compatible and has many improvements in speed compared to the earlier versions. There is a new manual in an A5 ring binder, which is more informative than previous manuals.

So why is The Painter special? On loading you soon notice the first difference. All the commands are chosen from menus on-screen. This option screen is actually separate from the main drawing screen – so that the screen does not become cluttered – and pressing 'ENTER' switches between the two screens. The number of different options seems rather daunting, but because the program uses Qjump's pointer interface, it is very simple to operate, and soon the manual becomes just something to refer to when using the more complex commands. There is a help function available from within the program which acts as a useful reminder on using the more complicated options.

The Painter appears to be a very professional program. It can be operated as easily from the keyboard or a joystick as from a mouse: a special configuration program allows you to specify either mouse settings or the speed at which the keyboard is read (if you do not possess a mouse). Users with full-size keyboards may experience a little difficulty in getting the settings so that the on-screen pointer does not move so fast as to be unusable, but with a little perseverance, the program can soon be set up to match your keyboard skills. I do not possess a mouse, but would love to see the program working with one, as the point trail mode (which leaves a trail of dots or a pattern on screen as

you move the pointer around) would be a joy to use.

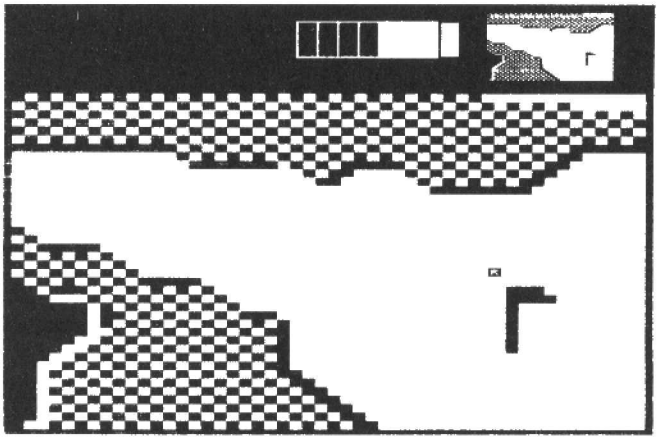
The program can deal with mode 4 and mode 8 screens (it can also convert them) as well as having all of the normal commands associated with drawing packages – circle, line, plot, square, fill, and many more. It does however also have some new and different commands for drawing pictures and for adding text to the screen.

On the drawing side, the standard QL ARC command has been replaced by a new 'Curve' command. This enables many more types of curves to be drawn. You simply choose the command from the menu screen, switch to the drawing screen (press ENTER), select the two ends of the curve and then use the cursor keys to move the high point of the curve.

There is also a 'Turn Shape' command which is actually more complex than its name may at first suggest. You need to outline the area to be 'turned' using a rubber-banded box. You can then specify four corners of any four sided shape into which the area will be moved. This can overlap the original, or be in a different place, but in any case, the transformation is smooth. It can even be used to re-shape a rectangle into a bow, for example. This command is very powerful, although it may also magnify/shrink the area which is to be re-shaped, and for this reason, I would have preferred the option to merely rotate a four sided shape as well.

Other graphics commands which are omitted by many of the other drawing packages include several methods of 'wiping' part of the picture, Pan/Scroll part of the screen (useful to line up parts of the picture exactly where you want them) and Zoom in/out (allows you to actually magnify part of the picture on screen if you decide that you drew something in the wrong size. All this takes place with a speed and ease of operation which leaves most programs well behind.

There is no need to worry if you make a mistake: if you press ESC after carrying out any of the commands, the screen immediately reverts to its previous state. Unfortunately this will only remove the last change to the picture, so for example, if you are drawing a series of lines, and decide to remove them altogether, you will have to use one of the Wipe options, which will also remove anything printed underneath the lines.



The same picture using the black and white screen jump option (here on a different print scale).

Colours on screen are easily chosen from a separate screen which allows you to either pick the colours and stipple manually, or from a large panel which shows all of the possible combinations. If neither of these suits your needs, the program can also use user-defined patterns which can be used to fill an area of the screen.

No mistake

When your drawing is nearly complete, the Screen Edit command comes into its own. This allows you to select a small area of the screen which is magnified so that it may be altered pixel by pixel. As you alter each pixel, a real-size representation of that part of the screen is shown in the top right corner so that you can see if the changes are having the desired effect. This is much quicker than the Magnified mode of GraphiQL Plus, although it does suffer a little by the fact that the pointer has to be moved to the top of the screen each time a different colour is needed. Unfortunately, this option cannot be used to line up the pointer on the overall screen, which would be useful, for example, to ensure that lines met each other.

Much thought has also gone into the text-manipulation commands, which allow text to be shown on screen in various ways. Not only can you use different fonts (standard QL fonts cannot be used, but there are several supplied with the program plus the option to design your own from within the program) and alter the character size, but you can also select to show the fonts in italics, outline only, bold, or even shaded (from four different angles).

There are two screens which allow you to specify how text is to be dealt with, the second of which allows even more complex operations. Text can be written not only sideways, but also on a slant and with a 3D effect (letters further along the slant appear smaller).

This text manipulation allows very complex screen displays to be created with writing in all possible directions, and in fonts up to 40x40 pixels in size (up to fifteen fonts may be loaded at any time).

The tools to help you produce a picture do not end here. The Painter can hold up to 12 different pictures at a time (depending on available memory), and parts of one picture can be lifted across to another screen using the filter and copy screen commands. The Filter command is useful if you set up a screen with several standard drawings on it, or just to doodle on. You could then opt to 'filter' out the background colour (eg black) which will enable this standard drawing to be placed anywhere on your current picture without affecting the background (useful for incorporating clipart).

Switching between the different screens couldn't be easier – you simply choose the View option, which then shows a small representation of all the open screens. You can then open or close a screen, move parts of one screen across to another, or simply opt to work on a different screen, all by moving the pointer using the cursor keys and pressing space. The multiscreen ability of The Painter must surely make it one of the most versatile drawing programs available for any home computer.

Once you have designed your screen, it can be saved to disk in normal 32K form or as a compressed screen. The compression routine is the best I have seen for the QL, although unfortunately at the moment there is no load-only version of the routine to allow people to use screens saved using it in their own programs.

If you want a hard-copy of the screen, an excellent screen-dump is included within the program. This can operate both 24-pin and 9-pin printers (it is apparently better with Epson compatibles), print the whole or part of a screen in three different sizes, print sideways, colour inverted, or black and white only (this turns off the shading used for the different colours).

There are many other options. Suffice to say that there are not many options which are not provided by this program (although I have yet to find a drawing package which can fill an area filled with a stipple colour).

There are no real problems with using the program (although some tv users may experience difficulty in reading some of the commands), however, despite it being written to run under Qjump's Pointer interface, oddly enough if you use a version of PTR_GEN later than that supplied with the program, you may end up with two pointers on screen, both moving about in response to the cursor keys!

As with any program, there are always one or two extras which the user adds to his/her wish list, but overall the program would seem to suit most people's drawing needs. It may seem a little expensive (mainly due to having to order it outside the UK), but I am certain that most users would say it was well worth it.

A QL COUSIN

The ICL One Per Desk is also a friend of David Warne's

The OPD (One Per Desk) computer was made by ICL Ltd. as a collaborative project between ICL, British Telecom and Sinclair with Psion providing the software. The same machine was badged for BT as the Merlin Tonto and for Australian Telecom as the Computerphone. The machine was intended for the busy executive with only limited computer skills. Most operations use multiple choice menus.

The hardware is based on the QL using the same 68008 processor, QL ulas, 128K of ram and microdrive data storage. The machine has a 'footprint' of about the same length but twice the depth of the QL.

Being launched shortly after the QL, the OPD suffered from the bad publicity attached to the microdrives, although ICL had much improved the reliability of the units. Despite this poor start, many hundreds were sold to local authorities, government departments and large companies. These are now coming on to the second hand market. They are regularly seen in magazines such as *Micro Mart* and *Computer Shopper*, with the cheapest units at about £100.00. Some computer supply companies are now replacing OPDs for their customers (mostly with PCs) and are selling the OPDs off cheaply.

A typical OPD featured a mono (black and white) monitor, twin microdrives, battery-backed clock, on-screen calculator, enhanced telephone handling, modem, Xchange software (*Abacus, Archive, Quill* and *Easel*), Basic and messaging (fax look-alike between OPDs).

The OPD is designed to be left on and the screen will blank if no keys are pressed for 10 minutes. Pressing any key, or an incoming call, re-activates the screen. The monitor is intended to be switched off between sessions leaving the computer powered for unattended functions. Monitors are available in 9 inch black and white or 14 inch Microvitec colour.

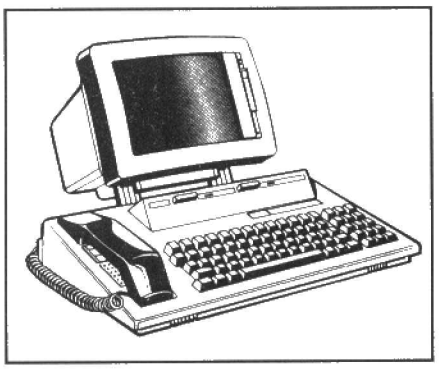
The microdrives are similar to those on the QL, but save the data in a different density. Although blank cartridges can be used on either machine, the OPD cannot read QL cartridge data. There is a program (for the QL) by Dave Walker of *DiscOver* fame, that can convert data and Basic from the OPD to the QL and vice versa. The OPD records cartridge use and read failures, and warns when the cartridge is due for renewal. The microdrives are very reliable.

The telephone has auto-dialing from a saved Telephone Directory/Address file of as many as 500 entries if required, with optional monitoring of cost and duration of calls. A running total is kept in memory. The directory has a search routine and short code dialling. There is a re-dial facility of any of the last six calls. Calls can be initiated through a built-in loudspeaker, the handset being picked up only when the connection is made. The machine will answer incoming calls using a programmable computer voice chip with different replies available for different times, eg lunch, holidays, gone home etc. There is, no facility to record incoming calls.

The modem is built-in and capable of Viewdata and Glass Teletype communications. This enables connection to Prestel, Yellow Pages, Tony Firshman's Board and many others. Screens can be saved to memory using the 'Snapshot' option, or entire programs can be downloaded to microdrive. Text can be prepared off-line to save phone charges.

The software is an enhanced suite of the programs supplied with the QL with the import/export of data between applications simplified. Being rom-based, it loads quickly and without read failures. The four are brought together under an operating 'shell' called Xchange Task Control. Up to eight 'tasks' can be in progress at one time. Import and Export between Psion programs is fast and simple. Xchange was an optional extra.

Basic is loaded from cartridge and is a reduced version of Superbasic. Many of the features of Superbasic are not available on the OPD. There are no graphics as such (CIRCLE, LINE, BORDER, FLASH etc.), no EXEC and no DIR. The screen size of the OPD is also slightly smaller. QL Superbasic programs can be transferred to the OPD but need considerable editing before use. Although using the same cpu, QL machine code programs cannot be run on the OPD. There are differences in the way the OPD handles the screen etc. that make QL programs incompatible.



Many OPDs were supplied with 'Messaging', a fax look-alike. This is in the form of a capsule that plugs into the back of the unit. The capsule contains a rom with the necessary code, which enables OPDs to send pre-typed text to each other using the telephone system. Received Messages can be edited and sent on to other users, printed or saved on microdrive. Later roms allow the messages to be sent at pre-set times and to different numbers to take advantage of cheap rate calls.

The OPD is provided with a serial port, but this works in one direction only, being intended solely for printer use. It is possible to download Basic files directly to the QL using the SER 2 port on the QL and a suitable cable. No input from the QL is possible by this route.

Later improvements included disk drives from PCML (with 256K extra memory), and another from Computer One, but these are no longer manufactured and can only be obtained on the second hand market. A variety of plug-in capsules were also provided, but most were to enable the OPD to connect to ICL mainframe computers and are of little use to enthusiasts. There were later options to allow the direct transfer of files direct from microdrive, via the telephone line, between OPDs and to import data into Quill or Abacus from Bulletin Boards.

It is not possible to simply plug-in extra memory as on the QL because the OPD requires special code to 'log-on and identify' the memory. A 128K expansion unit was made but few seem to have been sold.

Software is in very short supply, possibly because there is no organised user group such as Quanta, although some business oriented programs were produced. A diary/appointments program has been seen and also a CP/M operating system is available on one version of the disk drives. It is reported that Basic and 'C' compilers were produced but no sign of one has been seen.

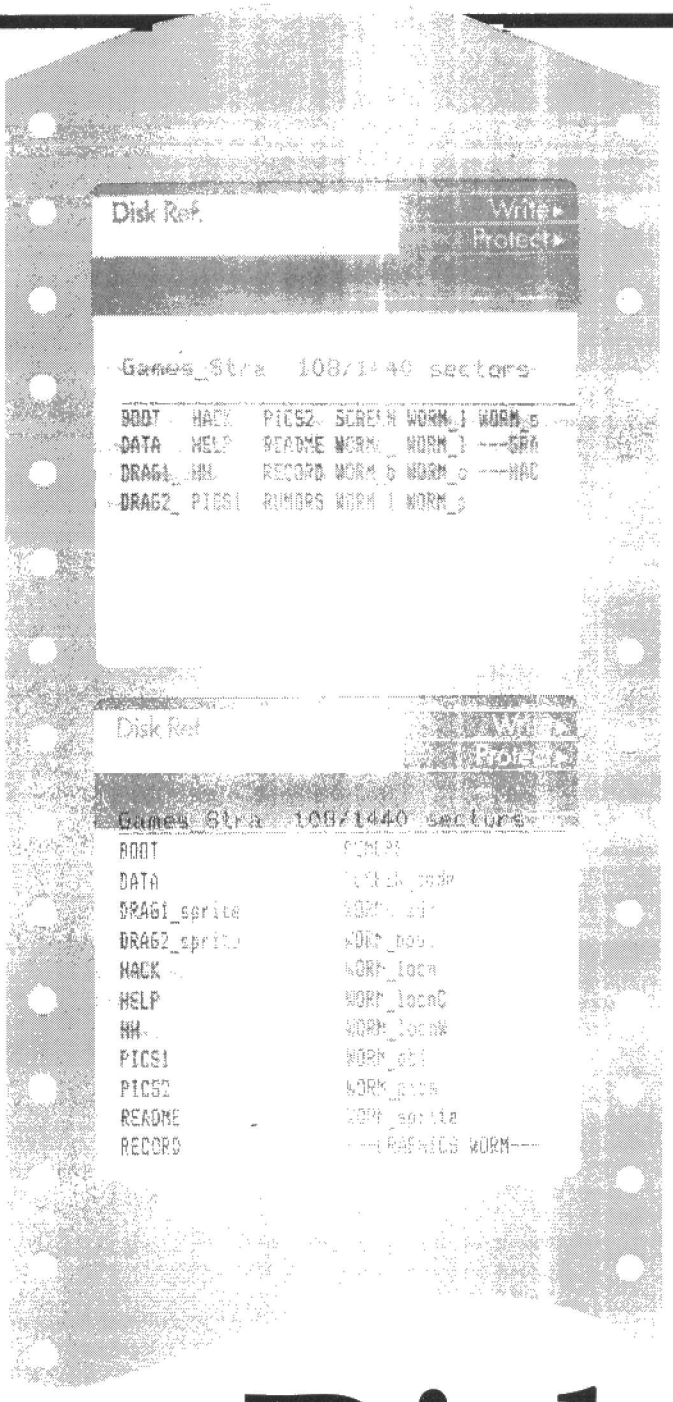
QL owners who use their machine mostly for the Psion package will find the OPD easy to use and in a business environment, a very useful tool. Sadly, it suffers from the lack of compatibility with the QL in the most important area, programs. Very little software is produced for the QL in Superbasic now and useful programs like *Flashback* and Tony Tebby's utilities cannot be used. The most promising area lies in Xchange applications and it is possible to transfer the *Archiver* group of programs once sold by Eidersoft.

I have both an OPD and a QL and swop data between them regularly. I like both of them. The OPD has made me friends in Scotland, Eire and Australia and the QL many at the local Quanta group.

INFORMATION
Program: *Super Disk Labeller*
Supplier: Dilwyn Jones Computing, 41 Bro Emrys, Tal-y-bont, Bangor, Gwynedd. Tel: 0248 354023.
Price: £10.00 (p&p included) for 3.5 and 5.25 disks or mdvs. 256K memory required.

Super Disk Labeller

John Shaw delves into the possibilities offered by a small package dedicated to sorting and labelling disk files.



Here we are then and just when I needed it. the program to make tidy printed labels for your floppy disks and what a good job it does. I have always wanted my Quanta library disks to have the contents listed on their labels and now I can do it.

This program can print labels that show the filenames on the disk in small print in several columns, plus a heading or title. It can also optionally include statistics such as file size, type and update date.

The reconfigurable printer driver with the package gives you control over the size of the text used and hence the amount of filenames that can be catalogued on the label.

Sorted

Filenames can be sorted into alphabetical order and even grouped so that, for example, only Quill_doc files are listed. Once the label has been assembled you can then edit the text, if required, or add your own text if there is room. There are quite a number of permutations possible.

There are also three special facilities.

Number one is the Quick Label facility which simply makes a sorted multicolumn filename label for the disk in a default drive with the minimum of key presses from the user.

Number two is a Sleeve Insert which is a list of filenames on a piece of paper larger than a label for keeping with the disk, especially if you use sleeves with your disks – mostly relevant to 5.25 in disks, although some 3.5 in disks are also supplied with clear plastic sleeves. With this facility, you can print a sheet showing filenames on a disk together with the disk name, which can be filed away easily.

Printer

Before using the program, you should select the printer driver to use with your printer. Several printer drivers are supplied with the program as follows:

DEFAULT_DRIVER: This is a very simple driver using very basic control codes that should work on just about any printer.

9 PIN_DRIVER This is a general purpose printer driver that seems to work with all Epson compatible printers that can print in condensed elite subscript print. It has been tried before release on Star LC10 and Centronics GLPII printers quite successfully.

9 PIN_PROP_DRIVER This is a proportional spacing printing version of the above.

24 PIN_DRIVER A 24 pin version of the nine pin driver above.

24 PIN_PROP_DRIVER A proportional spacing version of the 24 pin printer driver.

The program is driven by function key menus: F1 for select drive and read directory, F2 for sort, group and select filenames, F3 for make/view/edit label and backup, F4 for printouts and printer drivers and F5 for quick label. It is capable of reading what is on a drive, assembling a label and starting to print it after it has

sorted the filenames into order.

The printing will consist of a heading, made up of the disk title and capacity in sectors. All columns are the same size and the width of the column depends on the longest filename found on the disk.

The Sleeve insert facility enables you to print a piece of paper that can be stored with the disk in its sleeve, if you use them, or can be used in a disk box to remind you where each disk goes.

Selected

The Files Tidy List is a complete list of selected files on a disk, printed on a normal sheet of paper. It differs slightly in that it allows you to manipulate the list of files on a disk first, such as selecting only the _doc files, for example. The purpose is for you to be able to store on paper, for easy reference, a master list of files that you can look up at a later date. As it allows you to print on whole pages, you can use it for disks that have a very large number of files.

The Main Label making function is a little more complicated than the Quick Label function. To enable you to perform more manipulations on the label content before printing it, the filenames are read into an array list, where you can sort them into order, group by prefix (filenames starting with given letters or directory) and suffix (filenames with a certain extension only, such as _doc files).

Menu

On the same menu there are other processes that can be applied to the list. The first and most obvious is to sort it into alphabetical order. You can opt to group a list of files according to a suffix or extension name. For example, if you wanted all the doc files listed together and all the _dbf files listed together. Up to ten extensions can be listed for the program to do the matching.

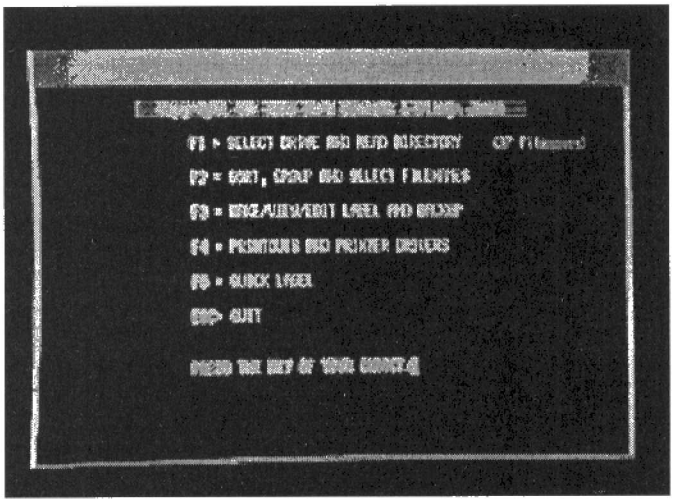
The program runs through the list of filenames, placing them in order as it finds the matching extensions. On the Super Disk Labeller disk, for example, you could list all the printer driver files together by using the extension _dat.

There is also a similar facility to group filenames by their root. Root in this case means the first few characters of a name, which could simply be a letter or two at the start of a name.

The final option on this menu allows you to undo any sorting or grouping operations that you may have performed on the files list. It simply restores the original list as it was read from the disk by killing the index list within the program.

Once you have sorted, grouped and selected all the files you want into the order you want, you then have to make a label. This is a copy in memory of what the label will be like when printed. After building up the label layout you can then preview and edit the lines of text.

The first two fixed types are the commonly available sizes of labels on a roll for 3.5 in disks. Type one is the larger of the two types. This is the type of label that folds round the edge of the disk. The second type is a label that does not wrap around the edge, but fills the label area on a 3.5 in disk.



"This program can print labels that show the filenames on the disk in columns and optionally include file sizes, etc."

Others

The 5.25 in labels are the third type. These are smaller labels that fit in one corner of the disk. This type will probably not be used much as most QL owners use 3.5 in disks. The fourth type is an address label, 3.5 inches wide and 1.5 inches deep. The final type calls on you to enter the label size yourself.

Next, you are asked for the text area layout details. Five choices are shown on the screen: single column, filenames only, single column, with statistics, multiple columns, with full filenames, multiple columns, best fit and squashed best fit. Once you have sorted, grouped and selected the list of files as you want to, there is one more special facility. The Backup utility is a short method of making a backup copy of the disk. The files are copied in the order held in the list in Super Disk Labeller. It can look very professional when all the filenames are listed in alphabetical order or grouped when looking at the disk with the DIR command in Basic.

There is a Customise Driver option that lets you edit existing printer drivers or create new ones specifically for your printer. There are some thirty sets of codes to enter, so you will need your typing fingers in good working order and plenty of patience.

Options

To give you some idea of the available options, a selection from the menus include: Group by Root, Group by Extension, Manual selection, Sleeve insert printout driver, Files tidy list printout, Select label types, Backup selected files, Customise printer, Use default printer.

So, here we have a superb piece of programming utility, containing all the possible options you could wish for and sold together with a 20-page User Manual which I found easy to use and follow.

In addition, if you have a very unusual printer, you can contact Dilwyn Jones and he will do his utmost to be of assistance.

A good, well thought out package, representing excellent value for money.

```
Games Stra  108/1440 sectors  (23 filenames)
BOOT            HACK            PICS2           SCREEN_code     WORM_locnC      WORM_sprite
DATA            HELP            README          WORMc_adv       WORM_locnW      ---GRAPHICS WORM---
DRAG1_sprite    HH              RECORD          WORM_boot       WORM_obj        ---HACK---
DRAG2_sprite    PICS1           RUMORS          WORM_locn       WORM_pics
```

Systemati
Machine Code
Programm

The first in a new series by Alan Bridewell, building on basic Machine Code skills.

I should begin by asking the questions: 'Why bother with machine code? Why not stick to Basic, or in the case of the lucky QL owner, stick to SuperBasic, surely the best implementation of the language on any machine?'

Up to a point, SuperBasic is fine. I write SuperBasic programs, and use other people's SuperBasic programs all the time, with no complaints. None, that is, until I need some rapid screen handling, or I need them to multi-task, or I'm sick of waiting forever for the program to load, or . . .

The problem is, the microprocessor in the QL (or any other computer) doesn't understand SuperBasic. So each command has to go through the interpreter, which converts it into numbers the microprocessor can understand. As a result, the QL spends most of its time trying to work out what the commands mean, and only a tiny fraction of its time actually carrying them out. Even worse, the microprocessor *never* learns, so no matter how many times your SuperBasic program loops through the same set of commands, they *always* need interpreting over and over again. What a waste of time!

The next question should be: 'What about compiled SuperBasic, using *Supercharge, QLiberator*, or *Turbo*?'

Compiler

A SuperBasic compiler will turn your SuperBasic program into a set of instructions the microprocessor can understand, so it doesn't have to wait for the interpreter, and can carry them out at once. It's magic to see your sluggish SuperBasic program become a fast, compiled program. And not only that, but it multi-tasks as well. What more could you want?

Well, there are ways, and ways, of telling a microprocessor what SuperBasic was designed to tell it to do. Compilers tend to be long winded. As a result, the program tends to get very long, and expensive in ram use. Not very useful if you want several multi-tasking in your limited ram space. Not only that, but often there are much faster sets of instructions than the compiler can find. So although compiled SuperBasic is much faster than interpreted SuperBasic, the program could be much faster still.

Which leads us to the question: 'What about other compiled languages?'

SuperBasic was designed to make things as easy as possible for human beings to tell the QL what to do. As a result, we get all the problems so far mentioned. Other languages, like Forth and C, are more of a half-way house. They are rather more difficult for us humans, but not *that* difficult, so most of us can pick them up if we're prepared to put the time and effort into learning them. The big advantage they have over compiled SuperBasic is that the compiler converts them into instructions that are more compact, and faster. But, even so, none of them can do it perfectly. The microprocessor can still be instructed in a more concise, and faster, way. But, in the end, the only way to do this is to instruct the microprocessor in its own language.

Unfortunately, the microprocessor's own language is an ever-changing series of numbers, which only computing masochists would attempt to use. But help is at hand, in the form of *assembler language*. This consists of a set of instructions which human beings can easily understand, but where each instruction converts to a unique number (or set of numbers) in the microprocessor's machine code language. For instance, in assembler language, we could write.

```
MOVE. L D4, D2
```

which simply means 'Copy the long word stored in the D4 register of the microprocessor into the D2 register of the microprocessor'. The assembler will convert this into the hexadecimal number $2404, which the microprocessor will recognise as that precise instruction, and writes it into the program. What could be easier? If some experts are to be believed, nothing is easier.

Well, in spite of what some of the experts say, it is a fact that the vast majority of computer users find machine-code programming very difficult, if not actually impossible. So, why should this be?

It's not that following the architecture of the microprocessor is difficult. Ideas like address, data, register and program counter are easy enough. And the lines of program are easy enough in themselves. Some of the instructions are quite subtle, but with a bit of thought they are all understandable.) With some effort, most of us who dabble in machine code can follow, say, Simon Goodwin's *DIY Toolkit* routines, and possibly make slight modifications to the code to suit our own needs. But to go from there to writing whole programs from scratch is not on.

The problems, I believe, come down to these.

1. It is too big a jump of imagination to go from single program lines to working out what large chunks of code will do. It's one thing to write code which simply moves numbers around registers and ram. It's quite another thing to work out how doing this can make things appear on the screen, or open files, or print to the printer, or any of the real jobs you want your program to do. We need small chunks of code which perform a recognisable task.
2. Many (possibly all) of these chunks of code are already there as trap calls or vectored routines. However to use them we have to put certain data in certain registers, and then sometimes look at other registers to find the result of calling the code. As it's almost impossible to remember where all this data has to go, it means continually referring to a m/c programming manual every couple of lines we write.

As a self-taught programmer, who does it for amusement, I have worked out a system which has enabled me to produce a number of useful programs.

What I have produced, and what I would like to share with you, is a programming kit consisting of small chunks of code to perform particular functions. They are fully annotated to show how they work, and how they may be fitted together with other chunks of code. In particular, special note is made of any alterations they might need

c
ing

in particular circumstances. For instance, every time the stack is used, the effect on the stack pointer is noted, so that when we need to use some data stored on the stack, we know where it is stored in relation to the current stack pointer. Also, if the code requires any data blocks, or buffers, or equates, these are set out so they can be conveniently moved to the end of the code to be lumped together.

At this point I should say that I program using *Assembler Workbench* by Talent. (You should have no problems modifying the listings to suit your assembler.) I learned most of my machine code work by reading Adam Denning's *Advanced QL Machine Code,* and a lot of what I have was produced by taking his programs apart and using the bits. I have also referred much to Adrian Dickens' *QL Advanced User Guide,* and, of course, to the many m/c programs that have been published over the years by *QL World,* and before that by *QL User.*

I should also point out that these articles are not designed as a beginner's tutorial in assembler language. (If you are not familiar with it you should obtain a suitable guide.) These articles are aimed at helping the many who can read programs to be able to write programs.

We will start by listing some of these chunks of code, and, where necessary, explaining what they do and how they do it. Then later, we shall put some of them together to make a program.

Jobstart (Listing one)

'Jobstart' is the 'standard' way to start the code for a multi-tasking job. I'm not sure it's actually necessary, but it does allow you to identify the job among others, using a toolkit command which prints out the state of the current jobs in the machine. (I don't know how it works - perhaps someday I'll find out.)

Priority (Listing two)

Often the first thing we want a job to do is to set its priority to fix what proportion of the processor's time it is going to use. (Basic has a priority of 32.)

'Priority' is a TRAP 1 call. We must put #MT_PRIOR in DO (#MT_PRIOR is the hex number B). We must put the job ID in

Listing 1

```
; ***************************************************************
;                          'JOBSTART'
; ***************************************************************
; THIS SHOULD BE HOW A MULTI-TASKING JOB SHOULD BE STARTED.
; START IS THE ADDRESS OF THE FIRST INSTRUCTION OF THE JOB.
; THE CHARACTER COUNT AND JOB NAME SHOULD BE EDITED AS APPROPRIATE.
;
                    BRA.S     START       ; BRANCH TO START OF CODE
                    DC.L      0           ; (THIS IS STANDARD FORMAT FOR
                    DC.W      $4AFB       ; START OF A JOB)
                    DC.W      4           ; CHARACTER COUNT OF JOB NAME
                    DC.B      'TEST'      ; NAME OF JOB
;
; START SHOULD NORMALLY POINT TO CODE SETTING THE PRIORITY OF THE JOB
; ***************************************************************
```

Listing 2

```
; ***************************************************************
;                          'PRIORITY'
; ***************************************************************
; THIS CODE SETS THE PRIORITY OF THE MULTI-TASKING JOB.
; THE VALUE IN D2 SHOULD BE ADJUSTED TO THE PRIORITY REQUIRED
;
.PRIORITY           MOVEQ     #$B,D0              ; #MT_PRIOR IN D0
                    MOVEQ     #-1,D1              ; OF THIS JOB
                    MOVEQ     #1,D2               ; TO 1
                    TRAP      #1
;
; ***************************************************************
```

Listing 3

```
; ***************************************************************
;                          'CONSOLE'
; ***************************************************************
; THIS IS HOW WE SET UP A CONSOLE WINDOW.
; MOST JOBS WILL REQUIRE AT LEAST ONE CONSOLE WINDOW.
;
; OPEN THE CONSOLE CHANNEL
;
                    LEA.L     PBLOCK,A1           ; PBLOCK ADDRESS IN A1
                    MOVE.W    $C6,A2              ; UT_CON VECTOR IN A2
                    JSR       (A2)
;
; SAVE THE CHANNEL ID WHICH UT_CON ROUTINE LEAVES IN A0.
;
                    MOVE.L    A0,-(A7)            ; SAVE CONSOLE ID ON STACK
                                                  ; * NOTE * THIS CHANGES A7
                                                  ; BY -4, READY FOR NEXT
                                                  ; STACK ENTRY.
;
; CLEAR THE WINDOW (OPTIONAL)
;
                    MOVE.W    #$FFFF,D3           ; INFINITE TIMEOUT
                    MOVEQ     #$20,D0             ; #SD_CLEAR IN D0
                    TRAP      #3
                    BRA.S     NEXTBIT             ; SKIP BLOCK
;
; * NOTE * NORMALLY, WE NEED TO PUT CHANNEL ID IN A0 BEFORE USING THIS
; TRAP TO CLEAR THE WINDOW. HOWEVER, IN THIS CASE IT HAS ALREADY BEEN
; LEFT THERE BY THE PREVIOUS SUBROUTINE
;
; DEFINITION BLOCK.  ADJUST VALUES TO FIT REQUIREMENTS
;
.PBLOCK             DC.B      4                   ; GREEN BORDER
                    DC.B      2                   ; 2 PIXELS WIDE
                    DC.B      0                   ; BLACK PAPER/STRIP
                    DC.B      7                   ; WHITE INK
                    DC.W      100                 ; WIDTH
                    DC.W      100                 ; HEIGHT
                    DC.W      0                   ; X POSITION
                    DC.W      0                   ; Y POSITION
;
; ***************************************************************
```

Listing 4

```
; ***************************************************************
;                          'MESSAGE'
; ***************************************************************
; THIS CODE WILL PRINT A STRING IN A WINDOW.
; THE BASE ADDRESS OF THE STRING MUST BE IN A1.
; THE CHANNEL ID OF THE WINDOW MUST BE IN A0. IF IT HAS BEEN LEFT THERE
; FROM THE PREVIOUS CODE, WE DON'T NEED TO DO ANYTHING ABOUT IT. (IF WE
; DO, WE'LL ONLY PUT A NUMBER WHERE IT ALREADY IS SO IT WON'T MATTER.)
; IF THE CHANNEL ID WAS THE LAST ITEM ON THE STACK IT WILL BE FOUND AT
; (A7), IF THE 2ND FROM LAST AT 4(A7), IF THE 3RD FROM LAST AT 8(A7), ETC
;
.MESS               MOVE.L    (A7),A0     ; OR 4(A7),A0 OR 8(A7),A0, ETC.
                                          ; CHANNEL ID IN A0
                    LEA.L     MESSAGE,A1  ; BASE ADDRESS IN A1
                    MOVE.W    $D0,A2      ; UT_MTEXT VECTOR IN A2
                    JSR       (A2)
                    BRA.S     NEXTBIT     ; SKIP MESSAGE
;
; MESSAGE. ADJUST LABEL, LENGTH AND CHARACTERS TO SUIT
;
.MESSAGE            DC.W      18          ; LENGTH OF MESSAGE
                    DC.B      'THIS IS A MESSAGE:'
```

D1. (In the case of the current job we put -1 into D1.) The actual priority we put into D2.

Console (Listing three)

Jobs almost invariably need at least one console window to input data from the keyboard. The routine 'Console' will open a console window, clear the window (optional), and place the console channel ID on the stack.

The routine uses the UT_CON vector which must be placed in A2 before jumping to its sub-routine. (UT_CON is $C6.) The subroutine will also require the address of a parameter block for the window in A1, and this must be in the following format:

00	byte	border colour
01	byte	border width
02	byte	paper colour
03	byte	ink colour
04	word	window width
06	word	window height
08	word	X origin (top left corner)
0A	word	Y origin (top left corner)

Message (Listing four)

The next thing we would normally want to do is to print some text to the window as a start-up message, or a prompt to input something from the keyboard. The routine 'Message' prints a message to the window.

It requires UT_MTEXT in A2, which it calls as a subroutine. (UT_MEXT is $DO). The subroutine requires the channel ID in A0, and the base address of the message in A1. This particular bit of code could possibly be used several times in a program, and each time it will need modifying. You need to adjust the label, length and characters of the message, and you will have to work out where the channel ID is stored in relation to the current stack pointer. (This only means carefully checking what has been added to the stack since the channel ID was stored.)

Instring (Listing five)

The routine 'Instring' is to input a string from the keyboard and place it in a store called Buffer. Here we run into a Qdos peculiarity which needs overcoming.

If we wish to use this string as a name, say, of a file we wish to open, then Qdos will expect the string to be preceded by a word holding the string length. However, the IO_FLINE routine we use to fetch the string just stores the string in buffer, and leaves the string length (including the LF at the end) in D1. So we precede 'buffer' with a one-word store 'BUF_POS'. We then put D1 in 'BUF_POS' and subtract 1 to remove the LF from the count.

Now, if we need the string preceded by the string count, we consider it as starting from 'BUF_POS'. But if we only need the string without the count, we consider it as starting from 'Buffer'.

Open (Listing six)

The routine 'Open' will open a channel using a name stored in 'BUF_POS'. (We use 'BUF_POS' rather than 'Buffer' be-

```
Listing 5

; ************************************************************************
;                              'INSTRING'
; ************************************************************************
; THIS INPUTS A LINE OF TEXT FROM KEYBOARD AND PLACES IT IN A STORE IN
; RAM CALLED 'BUFFER'.
; THIS STORE HAS TO BE LONG ENOUGH TO HOLD A LINE OF TEXT.
;
.INSTRING           MOVE.L    (A7),A0    ; OR 4(A7),A0 OR 8(A7),A0, ETC.
                                         ; CONSOLE CHANNEL ID IN A0
                    MOVEQ     #BUF_LEN,D2 ; LENGTH OF BUFFER IN D2
                    MOVE.W    #$FFFF,D3  ; INFINITE TIMEOUT
                    LEA.L     BUFFER,A1  ; BASE ADDRESS OF BUFFER IN A1
                    MOVEQ     #2,D0      ; #IO_FLINE IN D0
                    TRAP      #3
;
; IF THIS STRING IS TO BE USED BY QDOS AS, SAY, A CHANNEL NAME, THEN
; WE NEED THE ADDRESS OF CHANNEL NAME IN A0. QDOS EXPECTS THIS NAME TO BE
; PRECEDED BY A WORD HOLDING THE LENGTH. BUT IO_FLINE JUST RETURNS THE
; CHARACTERS, WITH A CHARACTER COUNT (INCLUDING THE LF) LEFT IN D1. WE
; MAKE SURE THE ADDRESS OF BUF_POS IS 2 LESS THAN THE ADDRESS OF 'BUFFER'.
; THEN WE SUBTRACT 1 FROM THE VALUE IN D1 AND PUT IT IN 'BUF_POS'. NOW
; PUTTING THE ADDRESS OF 'BUF_POS' FOR THE ADDRESS OF THE STRING GIVES US
; THE STRING IN THE CORRECT FORMAT.
;
;
                    LEA.L     BUF_POS,A0 ; BUF_POS IN A0
                    SUBQ.L    #1,D1      ; SUBTRACT 1 FROM THE D1 REMOVES
                                         ; THE LF FROM THE STRING COUNT.
                    MOVE.W    D1,(A0)    ; PUT NEW STRING LENGTH IN BUF_POS
;
; ************************************************************************
;                TO BE POSITIONED AFTER THE CODE
; WE NEED 'BUFFER' FOR THE STRING AND 'BUF_POS' FOR THE STRING COUNT
;
.BUF_LEN            EQU       100        ; LENGTH OF INPUT BUFFER
;
.BUF_POS            DC.W      0
;
.BUFFER             EQU       *
;
;          **** NOTE ****  BUFFER SHOULD COME LAST OF ALL.
;
; ************************************************************************
Listing 6

; ************************************************************************
;                              'OPEN'
; ************************************************************************
; THIS ROUTINE WILL OPEN A FILE
; WE NEED THE ADDRESS OF CHANNEL NAME IN A0. QDOS EXPECTS THIS NAME TO BE
; PRECEDED BY A WORD HOLDING THE LENGTH. SO WE USE 'BUF_POS' RATHER THAN
; 'BUFFER' AS THE POSITION OF THE START OF THE STRING SO IT STARTS WITH A
; STRING COUNT (SEE 'INSTRING')
;
                    LEA.L     BUF_POS,A0 ; BUF_POS IN A0
                    MOVEQ     #1,D3      ; OPEN OLD SHARED FILE, OR
                                         ; #0 = OLD EXCLUSIVE FILE
                                         ; #2 = NEW EXCLUSIVE FILE
                                         ; #3 = NEW OVERWRITE FILE
                                         ; #4 = OPEN DIRECTORY FILE
                    MOVEQ     #-1,D1     ; JOB ID FOR THIS JOB
                    MOVEQ     #1,D0      ; #IO_OPEN IN D0
                    TRAP      #2
                    TST.L     D0         ; ERROR?
                    BEQ.S     GOT_FILE   ; IF NOT, THEN CONTINUE. ELSE:-
; **** NOTE **** IF THIS ROUTINE IS REPEATED TO OPEN MORE THAN ONE FILE,
; THE NAME 'GOT_FILE' WILL NEED CHANGING FOR EACH OCCURANCE OF THE CODE.
;
; PRINT ERROR MESSAGE
;
                    MOVE.L    (A7),A0    ; OR 4(A7),A0, OR 8(A7),A0, ETC.
                                         ; CONSOLE CHANNEL ID IN A0
                    MOVE.W    $CC,A2     ; UT_ERR IN A2
                    JSR       (A2)       ; PRINT ERROR MESSAGE
;
; GO BACK TO SCREEN PROMPT FOR INPUT
;
                    BRA.S     MESSAGE    ; OR WHATEVER WE CALL THE CODE
;                                        ; FOR THE PROMPT.
;
; IF NO ERROR, THEN THIS CODE IS USED
;
.GOT_FILE           MOVE.L    A0,-(A7)   ; SAVE CHANNEL ID ON STACK
                                         ; * NOTE * STACK POINTER A7
                                         ; CHANGES BY -4
;
; * NOTE * AT THIS POINT THE NEXT CODE WILL PROBABLY BE TO PUT WINDOW ID
; BACK IN A0, TO USE THE WINDOW, OR MAYBE TO CLOSE IT.
;
Listing 7

; ************************************************************************
;                              'GETHEAP'
; ************************************************************************
; THIS ROUTINE WILL TRY TO ALLOCATE HEAP_ROOM OF COMMON HEAP SPACE AS A
; BUFFER. IF IT CANNOT ALLOCATE THIS MUCH, IT WILL HALVE THE NUMBER AND
; TRY AGAIN. IT WILL KEEP REPEATING THIS UNTIL IT ACTUALLY SUCCEEDS IN
; ALLOCATING AT LEAST 2 BYTES, OR GIVE UP. IF IT SUCCEEDS, IT WILL STORE
```

cause Qdos expects the string to be preceded by a word giving the string length.

We must take care to open the correct type of channel, and this is fixed by the value we put in D3.

0 = old exclusive
1 = old shared
2 = new exclusive
4 = new overwrite
5 = directory

If we are inputting data from an old file, then we would normally make it 'old shared', so other programs can use the file at the same time. But if we are outputting data to a new file, say a serial printer, or a screen window, we would normally make it 'new exclusive' so that it only gets data from this program.

As we might well use this routine more than once in a program, we must also be careful to alter the labels suitably each time it is used.

Getheap (Listing seven)

A useful thing is to use some of the spare, unused ram to speed up program by reducing the number of times we need to access a file on microdrive (or disk). So we allocate a chunk of the common heap (unused ram) as a buffer.

The simplest thing might be to work out (or guess) how much buffer we want, and allocate that amount of common heap for the use of the program. But since we don't know what other programs might be running at the same time, we don't know how much common heap is available. If there is not enough to give us what we ask for, the program will stop with an error message. So the best thing to do is to try to get the amount we want, but if there is not enough, we ask for less, and keep asking for less until Qdos gives us what we ask for.

The routine 'Getheap' keeps halving the amount asked for by using a LSR.L on D1, which contains the amount of ram we are asking for. (If no common heap is available, the program will just stop.)

Stringtr (Listing eight)

The routine 'Stringtr' will transfer strings of bytes from one channel to another. (Both channels will, of course, have to be already opened.) It will continue this until we get complete file transfer, or we get an error. The routine contains alternative bits of code depending on whether we use a small buffer at the end of the program to store the collected bytes, or whether we use some common heap. If we use a small buffer at the end of the program, we can probably use the same one we undoubtedly needed for storing the names of the channels we opened earlier. This means the last part of the routine is redundant, but has been put in for completeness.

This routine makes a lot of use of the stack, with three things stored there; the two channel IDs, and the error return when we input the string (so we can see if we

```
; THE ADDRESS AND NUMBER OF BYTES ALLOCATED IN HEAP_ADDR AND HEAP_LEN.
;
; NOW WE TRY TO GET THAT AMOUNT OF RAM FROM THE COMMON HEAP
;
                    MOVE.L    #HEAP_ROOM,D1 ; BYTES REQUIRED IN D1
.GET_ROOM           MOVEQ     #-1,D2    ; ID FOR THIS JOB IN D2
                    MOVEQ     #$18,D0   ; #MT_ALCHP IN D0
                    TRAP      #1        ; TRY TO GET RAM
                    TST.L     D0        ; ERROR
                    BEQ.S     GOT_AREA  ; YES, THEN CONTINUE. ELSE:-
                    LSR.L     #1,D1     ; DIVIDE BYTES REQUIRED BY 2
                    CMPI.L    #1,D1     ; IF D1<=1 STOP WITH ERROR
                    BGT.S     GET_ROOM  ; ELSE LOOP AND TRY AGAIN
                    BRA.S     JOB_END
;
; HAVING GOT SPACE ALLOCATED, WE MUST SAVE ADDRESS AND NUMBER OF BYTES
; ALLOCATED.
;
.GOT_AREA           LEA.L     HEAP_LENGTH,A1
                    MOVE.L    D1,(A1)   ; SAVE NUMBER OF BYTES
                    LEA.L     HEAP_ADDR,A1
                    MOVE.L    A0,(A1)   ; SAVE AREA ADDRESS
                    BRA.S     NEXTBIT   ; SKIP HEAP ALLOCATION BLOCK
;
; HEAP ALLOCATION BLOCK.
; FIRST WE NEED TO SAY HOW MUCH HEAP_ROOM WE WANT. 4096 BYTES IS 8 MDV
; SECTORS, SO WILL REQUIRE ONE MDV ACCESS FOR EVERY 8 MDV SECTORS IN
; THE FILE.
;
.HEAP_ROOM          EQU       4096      ; HEAP ROOM REQUESTED
.HEAP_ADDR          DC.L      0         ; SPACE FOR HEAP ADDRESS
.HEAP_LENGTH        DC.L      0         ; SPACE FOR LENGTH OF HEAP SPACE
;
; ***************************************************************
```

Listing 8

```
; ***************************************************************
;                        'STRINGTR'
; ***************************************************************
; THIS ROUTINE TAKES A STRING OF BYTES FROM A CHANNEL AND STORES THEM IN
; A BUFFER, UP TO THE LENGTH OF THE BUFFER (BUF_LEN). IT THEN SENDS THE
; STRING TO ANOTHER CHANNEL. IT THEN KEEPS REPEATING UNTIL WE GET EOF.
;
; FIRST WE FETCH THE STRING FROM THE INFILE
.FILE_P             MOVE.L    (A7),A0   ; OR 4(A7),A0, OR 8(A7),A0, ETC.
                                        ; INFILE_ID IN A0
                    MOVEQ     #BUF_LEN,D2 ; IF 'GETHEAP' USED, THEN
                                        ; LEA.L HEAP_LENGTH,A1
                                        ; MOVE.L (A1),D2
                                        ; LENGTH OF BUFFER IN D2
                    LEA.L     BUFFER,A1 ; IF 'GETHEAP' IS USED, THEN
                                        ; LEA.L HEAP_ADDR,A1
                                        ; MOVE.L (A1),A1
                                        ; BUFFER ADDRESS IN A1
                    MOVE.W    #$FFFF,D3 ; INFINITE TIMEOUT
                    MOVEQ     #3,D0     ; #IO_FSTRG IN D0
                    TRAP      #3        FETCH STRING
                    MOVE.L    D0,-(A7)  ; SAVE ERROR RETURN ON STACK
                                        ; * NOTE * STACK POINTER A7
                                        ; CHANGES BY -4
;
; NEXT WE SEND STRING TO OUTFILE
;
                    MOVE.L    4(A7),A0  ; OR 8(A7),A0, ETC.
                                        ; OUTFILE_ID IN A0
                    LEA.L     BUFFER,A1 ; IF 'GETHEAP IS USED, THEN
                                        ; LEA.L HEAP_ADDR,A1
                                        ; MOVE.L (A1),A1
                                        ; BUFFER ADDRESS IN A1
                    MOVE.L    D1,D2     ; IO_FSTRG LEAVES THE NUMBER OF
                                        ; BYTES ACTUALLY FETCHED IN D1
                                        ; SO WE PUT THIS IN D2
                    MOVEQ     #7,D0     ; #IO_SSTRG IN D0
                    TRAP      #3        ; SEND STRING
;
; AT THIS POINT WE NEED TO KNOW IF WE HAVE REACHED THE END OF THE FILE.
; SO WE EXAMINE THE ERROR RETURN SAVED ON THE STACK.
;
                    MOVE.L    (A7)+,D0  ; PUT ERROR RETURN BACK IN D0
                                        ; * NOTE * STACK POINTER A7
                                        ; CHANDES BY +4
                    TST.L     D0        ; ERROR?
                    BEQ.S     FILE_P    ; NO, THEN LOOP FOR NEXT STRING
;
; ***************************************************************
;                      POSITION AFTER THE CODE
; USE GETHEAP FOR A LARGE BUFFER IN THE COMMON HEAP AREA
; THIS BUFFER WILL HAVE ITS ADDRESS STORED IN HEAP_ADDR AND ITS LENGTH
; STORED IN HEAP_LEN
; OR....
; .BUF_LEN          EQU       100
; .BUFFER           EQU       *
; *** NOTE *** BUFFER SHOULD COME AT THE END
;
; ***************************************************************
```

have reached the EOF). We must be careful that we know exactly where the stack pointer is at each stage, otherwise we shall be taking the wrong data from the stack!

Close (Listing nine)

Whenever we finish with any channels it is important we close them. If the stack contains other data besides the channel ID we need, then we may have to be a bit careful here. However, if the program is not too complicated (and well ordered) usually by the time we are ready to close channels, all we have left on the stack are IDs of unwanted channels, so there is no problem.

Since closing channels is usually (though not always) followed by killing the job, I have included the line which will jump to the routine to do this.

Endjob (Listing ten)

'Endjob', the last routine in this first article, is the one needed to kill a job, either because it has been completed successfully, or because an error has occurred.

Fatal errors, when detected, should jump to JOB END, and this will give an appropriate error message in channel 0 before killing the job.

A successful completion should jump to END JOB.

In case you feel, after reading this first article, that Assembler Language programming is too difficult, I must point out that later parts are easier. Unfortunately, we can't really start to have a usable program without opening and closing channels to screen, printer, file, etc., and inputting and outputting text. So although these are not the easiest of jobs to program, they really need to come first if we are quickly to get to write a usable program.

Obviously, we will need much more than this if we are going to get very far with our programming. But these are enough for a start, and next time we shall use them to produce a successful program.

```
Listing 9

; ************************************************************************
;                                'CLOSE'
; ************************************************************************
; THIS ROUTINE WILL CLOSE AN OPEN CHANNEL WHOSE ID IS TOS
;
.CLOSE              MOVE.L    (A7)+,A0  ; CHANNEL ID IN A0
                    MOVEQ     #2,D0     ; #IO_CLOSE IN D0
                    TRAP      #2
;
; IF THIS IS THE LAST CHANNEL TO CLOSE BEFORE WE KILL THE JOB THEN THE
; NEXT LINE IS NEEDED TO KILL THE JOB. OTHERWISE, LEAVE OUT NEXT LINE.
;
                    BRA.S     END_JOB
;
; * NOTE * IF THE CHANNEL ID WE WANT TO CLOSE IS NOT TOS, WE SHALL HAVE TO
; REMOVE THE OTHER ITEMS ONTO REGISTERS (OR RAM) TO GET TO THE ITEM WE
; WANT, USE IO_CLOSE, AND THEN PUT THEM BACK ON THE STACK.
;
; ************************************************************************
Listing 10

; ************************************************************************
;                               'ENDJOB'
; ************************************************************************
; THE FIRST PART OF THIS ROUTINE SHOULD ONLY BE USED WHERE AN ERROR IS
; DETECTED BEFORE A SUITABLE WINDOW HAS BEEN OPENED. IT WILL GIVE AN
; ERROR MESSAGE IN CHANNEL 0, OR IN CHANNEL 1 IF CHANNEL 0 IS NOT
; AVAILABLE.
;
.JOB_END            MOVE.W    $CA,A2 ; UT_ERRO in A2
                    JSR       (A2)
;
; THE NEXT PART IS THE NORMAL ENDING OF A JOB BY KILLING IT.
;
.END_JOB            MOVEQ     #5,D0     ; #MT_FRJOB IN D0
                    MOVEQ     #-1,D1    ; ID OF THIS JOB IN D1
                    TRAP      #1
;
; ************************************************************************
```

DIY TOOLKIT

Simon Goodwin shows how you can turn QL task files into the keywords of your dreams.

Task Commander turns Qdos task files into resident procedures. Tasks run instantly when you type your chosen commands. This is not just a great way to extend SuperBasic; it also shows how to write code that creates, loads and activates QL tasks.

The program uses extended SuperBasic to add a header, giving a task a name, like a built-in or Toolkit command. The resultant code remains a fully-fledged task, so you can launch and run it at the same time as Basic, perhaps passing channels or other parameter data for it to work on.

All *Task Commander* files can run from rom, regardless of the compiler used to generate the task. The task code is copied into the transient program area before running. Tasks can be up to one megabyte in size; even the largest should start in under a second.

As usual, the programs are tested on Sinclair JM roms, *Minerva* Qdos 1.64 and Argos 6.41. As far as I know, files should be totally compatible between rom versions.

I have often been asked for some way to turn compiled tasks into keywords. At last I've done it, and my systems are now rapidly filling with useful new keywords as a consequence. I took my cue from Quanta boss Phil Borman, who kindly sent me a disk of his program MAKE_KEYWORD_BAS earlier this year. It uses Turbo Toolkit to add a short header to *Turbo* tasks.

Resulting files can be loaded into memory like any Toolkit command. The chosen keyword starts the task; for instance, you an type BLAST instead of EXEC_W FLP1_BLAST.

Phil's routine works well, but has limitations. The calling task waits until the new task is finished; there is no way to pass parameters, and keyword names cannot exceed seven characters or Name Table corruption might result.

The DIY Task Commander needs a little more ram than Phil's MAKE_KEYWORD. It is safe with any length of name, launches tasks even more quickly, and allows full Qdos task parameters. It can start a task from SuperBasic without stopping the interpreter. Just add a comma at the end of the command, to get the effect of EXC rather than EXEC_W.

Parameter passing between tasks was not mentioned in the original 1984 Qdos manual, but it is documented in section 24 of QJumps *Supertoolkit 2 Guide*. The standard method passes any number of channels on the task's stack, indicated by a count word followed by the corresponding number of long word channel identifiers. It can also pass a string of up to 32766 bytes.

This parameter-passing scheme allows pipes and filters, in the style of the Unix operating system. It is implemented by many QL compilers, including *GST C, Pro Pascal, Qliberator* and *Turbo*.

For instance, QLib assigns channels to #0, #1, #3, #4 and so on; Turbo tasks assign channels in order downwards from #15. Compiled programs read the string parameter by refering to a pre-assigned name like ARGV, CMD$ or OPTION_CMD$. Your compiler manuals should have details.

It appears that this scheme was intended from the start, as even the AH rom CJOB routine puts two zero words on the stack by default; these signify no channels and no characters.

The first half of **Listing one** evaluates, checks and passes the parameters on to the task. It is important to validate them all before allocating memory for the task; we don't want to waste time or memory if an error is inevitable.

The parameters are simple: any number of open SuperBasic channel numbers, followed by an optional string. The last parameter is taken as a string unless it has a # prefix; thus you can coerce a string to a channel number, and pass nothing but channels, or pass any number, coerced to a string.

In five years DIY Toolkit has demonstrated many styles of QL parameter scanning, allowing options, defaults, and the full range of data-types; now I'm going to show you how to do it backwards! Listing one scans from right to left. This is the easy way to pick up the comma and string parameter which may be at the end of the statement. Everything else must be channels.

This sequence means that the SuperBasic value-fetching routines put the parameters in the ideal order on the RI stack, so they can be copied *en bloc* to the data area of the task before it is activated. There may be anything from zero to a dozen or more parameters; typically you might pass a short string and zero, one or two channels, like this:

```
CHURN "flp1_help/flp2_report",
SPLICE #0,#3,"temp_merge"
```

If no explicit channels are specified, the DIY code passes the identifier of the Default SuperBasic channel #1. The task does not have to use this, but may find it convenient. The DIY Toolkit USE keyword, from February 1988 and Volume C, can make this the default channel in the compiled task, too; just add USE 0 for Qlib or USE 15 for Turbo.

The identifier of channel #1 upsets Psion *Xchange,* which does not expect to be passed any channels, and gives a bad parameter error otherwise. To fix this, change the first and last numbers in line 870 of **Listing two;** replace 26696 with 28232, and change 26122 into 24586 at the end of the line. This swaps BNE.S for BRA.S, so that #1 is not passed unless it is an explicit parameter.

Remember that Qdos cannot allow one task to use a channel while another is waiting for data from the same channel. It is unwise to pass SuperBasic windows #0,#1 or #2 to tasks if you intend to use the trailing comma to allow SuperBasic to carry on; output may be delayed or muddled. Programmers can check if a channel is busy by calling CHBASE, from DIY Toolkit, Volume Q.

The machine code in Listing one demonstrates some tricky parameter handling, and shows how tasks are created and activated. Listing one is the assembler source of the code added to each task, tested with *Devpac*.

You can type the code of Listing one into any QL assembler, and customise it for

your needs. Minor syntax variations mean you may need to suffix 'A' to a few address instructions, and omit '\w' when reading word vectors.

The Basic loader configures the code to suit each task, inserting the name and address of the Resident Procedure at PARAMS, patching the code and data sizes at MAKE_TASK, and tacking the code of the task onto the end, at TASK_CODE.

Task Commander is very useful for small tasks, unless you have plenty of memory. Some program tasks modify their code after loading, so you need two copies in memory to run one. The master copy needs no data space, but about 280 bytes are required to process the name and parameters, and launch the task.

QL tasks are small compared with memory capacity; most packages work in under 90 K, to suit the original QL, yet the long-established standard expansion of 640 K ram has been bettered by the 896 K *Trump Card*. Now the standard is set by the two megabyte *Gold Cards*, multi-megabyte Thors and Qdos emulators for ST and Amiga

Memory is cheap now, be it rom or ram. Task Commander is best if you have ram to spare, or itch to fill one of Quanta's 192 K Eprom expansion boards, or the 32 K rom sockets on CST's RAM+.

If all the tasks you use are re-entrant you could conserve memory by passing the address of TASK_CODE to MT.CJOB in A1, setting D2 to zero. This won't suit all compilers, but it saves making a copy of the code each time you want to run a task.

The code labelled ADJUST_A3 prevents the parameters being copied if there is not enough room for them in the data space.

Notice that the SUB.W D0,A3 instructions implicitly extend D0 to a long-word value before subtracting it from A3. This is important, as A3 invariably exceeds 32767. Two steps are used as D0 is a count in words, and A3 addresses bytes.

Assembler programmers soon learn the importance of keeping values in processor registers; this policy avoids repeated access to memory, and is vital for fast QL code. When there are many variables in use it may be difficult to assign registers.

As usual, the innermost loop is the one to optimise first. Values therein are used most, so they must be the most accessible. But which registers can be used? This is my approach.

Aim to preserve useful register values for your code and the routines it calls. The rom uses A0 for the Channel ID, D1 for the Task ID, D3 for timeout; other defaults were listed in July's DIY Toolkit Volume T. There are 16 general-purpose registers, rather than two or three of early processors; it is still easy to run out, especially when writing SuperBasic extensions.

A7 and A6 are pre-defined, delimiting the subroutine stack and the start of Basic memory; A1, A3 and A5 point at parameters, while A0 and A2 get clobbered by the CA.GT parameter routines, along with D0, D1, D2, D3, D4 and D6. The only spare registers are A4, the interpreters's pseudo-program counter, D5, and D7, which must sometimes be zeroed to avoid a maths bug in early Sinclair roms.Task Commander uses D5 and D7 to pass parameter details to the loader. D5.L and D5.W are used for two distinct purposes. The sign of D5.L distinguishes EXEC from EXEC_W, while D5.W holds the length of the parameter string.

When it gets tricky to allocate optimal register numbers, I find changes are kept to a minimum if I program using names like D8 and D9 as well as the 'real' registers D0 to D7. These must be replaced before the program is assembled, but they help me see just what is needed, and where.

Qdos saves and restores 68008 register values in the job header, immediately before the code space, as it swaps from task to task. The instruction before ACTIVATOR sets JB.A7, the stored value of the

```
DIY TOOLKIT - OCTOBER 1991 - Listing 1, page 1 of 2

* QL WORLD DIY TOOLKIT - TASK COMMANDER ROUTINES
* Version Ø.9, Copyright July 1991 Simon N Goodwin
*
* BP.INIT expects a PROC name & address to be inserted at PARAMS
*
setup       lea.l    params,a1        Point at the details
            move.w   $110\w,a2        Fetch BP.INIT vector
            jmp      (a2)             Add a resident procedure
params
*
not_open    moveq    #-6,dØ           CHANNEL NOT OPEN error code
bad_return  rts
*
* TASK_CMD_NAME { # CHANNEL% , }   [ PARAMETER$ ]   [ , ]
*
taskcmd     moveq    #Ø,d7            Default channel count
            moveq    #Ø,d5            Default parameter string: ""
            cmpa.l   a3,a5            Check parameter pointers
            beq.s    make_task
            btst     #4,-7(a5,a6.l)   Is there a trailing comma?
            beq.s    checked
            bset     #31,d5           Aha, it's EXEC not EXEC_W
            tst.b    -8(a5,a6.l)      Is there a data type too?
            bne.s    checked
            subq.l   #8,a5            Forget the last 'parameter' ,
            cmpa.l   a3,a5            Is there anything else?
            beq.s    make_task
checked     tst.b    -7(a5,a6.l)      Does the last one have a # ?
            bmi.s    get_chans        Positive if not, assume $
*
* Pick the string parameter off the end of the parameter list
*
get_string  movea.l  a3,a4            Note location of parameters
            lea.l    -8(a5),a3        Focus on the last one
            movea.w  $116\w,a2        Pick up CA.GTSTR vector
            jsr      (a2)             Get string onto RI stack
            bne.s    bad_return
            move.w   Ø(a1,a6.l),d5    D5 is text length, Ø-32767
            bne.s    find_chans
            addq.l   #2,a1            Ignore nulls on the RI stack
find_chans  cmpa.l   a3,a4            Is there anything else?
            beq.s    make_task
            movea.l  a3,a5            Find the other parameters
            movea.l  a4,a3            Recall pointer to first
get_chans   movea.w  $118\w,a2        Fetch CA.GTLIN vector
            jsr      (a2)             Stack some long integers
            bne.s    bad_return
            move.w   d3,d7            Update channel count
*
* Check each channel number before allocating task memory
*
            movea.l  a1,a4            Keep RI stack base offset
check_chan  move.w   2(a1,a6.l),dØ    Get LOW word of parameter 1
            bmi.s    not_open
            mulu     #4Ø,dØ           Each table entry is 4Ø bytes
            add.l    48(a6),dØ        Add channel base offset
            cmp.l    52(a6),dØ        Check it is within the table
            bge.s    not_open
            move.l   Ø(a6,dØ.l),dØ    Pick up the Qdos channel ID
            bmi.s    not_open
            move.l   dØ,Ø(a1,a6.l)    Store ID in place of #number
            addq.l   #4,a1            Advance to next parameter
            subq.w   #1,d3            Process all in turn
```

stack pointer in the job header. The default value set by MT.CJOB points at two zero words at the end of the data space; we need to move A7 down to address the parameters.

The main Task Commander program is written in SuperBasic. A version based on SuperToolkit extensions appears in Listing two. It merges the task with the DIY set-up code, adding the chosen command name. It uses common Toolkit commands, and can easily be converted; I have versions that use DIY Toolkit, Turbo Toolkit and SuperToolkit commands.

All three versions – plus documentation, examples and assembler source – have been added to the File Tools disk, Volume F of DIY Toolkit. Send £7 for the complete volume, which includes the *Customkit* toolkit file combiner, and a utility to create eprom headers. make payment out to **DIY Toolkit** (not to CGH Services), **Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA,** or phone Richard on (0559) 384574 1 pm - 9 pm.

There are now 17 volumes. Each costs £3 plus £4 per order to cover disks, post and processing. Please send one formatted cartridge per volume if you need the files on microdrive.

Listing Two uses a handful of Supertoolkit commands. Turbo Toolkit fans may prefer to replace FTEST and FOP_IN with DEVICE_STATUS; ALCHP and RECHP correspond to ALLOCATION and DEALLOCATE, while FDAT and FLEN mimic DATASPACE and POSITION at the end of the file.

The DIY version uses GetHEAD, RESERVE and DISCARD extensions, supplied with Volume F along with the source for both other versions. The differences stem from variations in the commands and error-trapping facilities of each toolkit.

The Turbo-Toolkit variant has the most comprehensive checks, but may stop if you run out of memory or space on your drive, or try to write to a 'read only' medium.

Messages appear if the source file, containing the task, is absent, too long or an odd length. Odd length files crash EXEC. The code copies 16 bytes at a time, in up to 65536 steps, moving up to a megabyte at high speed.

The task data space must be at least 16 bytes; most tasks use far more, so that the message 'data space too small' probably means your source is not a task file at all. Parameters need at least eight bytes, and the fast copier may overrun by up to 16 bytes when copying the task code; it's faster that way.

The destination file contains the new command code, which you can link in the usual way with LRESPR, LINKUP or RESPR, LBYTES and CALL. It needs slightly more space than the source file.

The code loader is not the usual DIY Toolkit routine; it does not make a temporary file, and stores decimal words rather than hex strings. DATASTORE is shorter, faster and more reliable than the hex loader which Marcus Jeffrey bequeathed to this column in 1987.

Each DATA line starts with a checksum, followed by a count of the number of 16 - bit values thereafter. The checks for each line detect extra or missing items and report the line number if a problem is found, so you need only correct that line. The exception is 'DATA length error', which indicates that the total of the 'count' values does not add up to the expected number of words. If you get this report, check the second value in each DATA line.

All the data is numeric, as many people have number key-pads and I wish to avoid confusion between the letter B and the digit 8. The digit zero appears with a slash throughout, to distinguish it from letter O.

Users with their own hex editor might prefer the hex representation. It is easy for me to use either format, as I have software tools to create and check the DATA, so let me know what you prefer, care of *QL World*.

You may need to comment out the RESTORE at line 600 before compiling Listing two. It is only needed when you first

```
DIY TOOLKIT - OCTOBER 1991 - listing 1, page 2 of 2
* D7.W is No. of channels; D5.W is text length; D5.L<Ø for EXEC
*
make_task   move.l    #Ø,d3             Data space - to be patched
            move.l    #2,d2             Code space - to be patched
            moveq     #-1,d1            Owner is this task
            suba.l    a1,a1             A1 is Ø; put task in TRNSP
            moveq     #1,dØ             MT.CJOB trap key
            trap      #1                Create a task
            tst.l     dØ                Did that work?
            bne.s     return_err
            lea.l     8(aØ,d2.l),a4     A4 -> start of data space +8
            lea.l     -8(a4,d3.l),a3    A3 points past end of dspace
*
* Copy the code into the task area; ignore slight over-runs
*
            lsr.l     #4,d2             Divide by 16 (roughly)
            move.l    aØ,a1             Copy pointer to code space
            lea.l     task_code,a2      Find the code
copy_code   move.l    (a2)+,(a1)+       Repetition improves speed
            move.l    (a2)+,(a1)+
            move.l    (a2)+,(a1)+       Move 16 bytes per DBRA step
            move.l    (a2)+,(a1)+
            dbra      d2,copy_code      Copy up to 1 Megabyte, fast
*
* Copy the parameters from RI to JB.SP, counting in 16 bit words
*
            move.w    d7,dØ
            add.w     dØ,dØ             Number of channel ID words
            tst.w     d5                Check string length
            bne.s     string
no_string   clr.w     -(a3)             Stack a null length
            bra.s     adjust_a3
string      addq.w    #3,d5             Include length + an odd byte
            lsr.w     #1,d5             D5 := string size in words
            add.w     d5,dØ             DØ := stack words required
adjust_a3   sub.w     dØ,a3             Move A3 down to make room
            sub.w     dØ,a3             DØ is in words, A3 is bytes
            cmpa.l    a4,a3             Ensure room for parameters
            bcs.s     activator         Pass nowt if they won't fit
            movea.l   a3,a4             Save bottom address
            move.l    $58(a6),a1        Recall the BV.RIP offset
            bra.s     copy_loop         Copy nothing if DØ=Ø
copy_pars   move.w    Ø(a1,a6.l),(a3)+
            addq.l    #2,a1             Advance up the RI stack
copy_loop   dbra      dØ,copy_pars
*
* If CHANS=Ø, stack 1.W and the ID from #1, else stack CHANS.W
*
chan_check  tst.w     d7                Check channel count
            bne.s     stack_ok
            moveq     #1,d7             Pass default, ID of #1
            movea.l   48(a6),a2         A2 is offset to BV.CHBAS
            move.l    4Ø(a6,a2.l),-(a4) Presume that #1 exists
stack_ok    move.w    d7,-(a4)          Stack the channel count
            move.l    a4,-12(aØ)        Set JB.A7 in task header
*
* Activate the task
*
activator   moveq     #16,d2            Default priority (Ø-255)
            moveq     #-1,d3            Timeouts Ø = EXEC, -1 = WAIT
            tst.l     d5                Did command end with a comma?
            bpl.s     time_set          If not, timeout is -1, wait
            moveq     #Ø,d3             EXEC; no need to wait
time_set    moveq     #1Ø,dØ            MT.ACTIV trap key
            trap      #1
return_err  rts                         Return error code (if any)
*
task_code   end                         Task code follows here
```

run the program to check the data, and Turbo or Supercharge object to the variable parameter. Once the data has been checked all you need is a single RESTORE at the start of the program.

After just a few weeks I find the Task Commander is indispensable. The compiled task, TASCOM, is itself a resident command on my QL and Thor; every new task is a candidate for installation. I am grateful to Phil Borman for sending me down this track.

The scope for new DIY Toolkit extensions seems to be narrowing, and I welcome new directions. Ideal topics involve short, original machine - code routines which can be used as 'building blocks' in other programs. Please send your suggestions to me.

```
DIY TOOLKIT - OCTOBER 1991 - listing 2, page 1 of 2

100 REMark Program to convert any Task into a Resident Procedure
110 REMark SuperTOOLKIT 2 version 1.1, with full parameter passing
120 maxcode=2^20 :REMark DIY fast load limits tasks to 1 Megabyte
130 :
140 WINDOW 400,150,36,30 : BORDER 2,0,7 : PAPER 0 : CLS
150 PAPER 2 : INK 7,4 : CSIZE 2,1 : CLS 3
160 PRINT "TASK COMMANDER QL file converter"
170 PAPER 7 : INK 2 : CSIZE 1,0 : CLS 3
180 PRINT "    Written by Phil Borman & Simon N Goodwin."\\
190 INK 4 : PAPER 0
200 :
210 INPUT ' Source file >'!source$
220 INPUT ' Destination >'!destin$
230 INPUT ' Keyword     >'!name$\\
240 IF name$="" : PRINT " Huh?" : STOP
250 flag=FOP_IN(source$)
260 IF flag<0 : PRINT ' Error reading'!source$ : REPORT #1,flag : STOP
270 file_length=FLEN(#flag)
280 file_data = FDAT(#flag)
290 CLOSE #flag
300 IF file_length>maxcode OR 2*INT(file_length/2)<>file_length
310   PRINT source$!"has odd or excessive size" : STOP
320 END IF
330 IF file_data<16 : PRINT !source$!"data space too small" : STOP
340 buffer=ALCHP(file_length+280+LEN(name$)) : NL%=LEN(name$)
350 LET b=buffer : DATASTORE 740 :REMark Store BP.INIT code
360 offset=6+NL%-(NL% && 1) :REMark Find the PROC code
370 POKE_W b,(NL%+8) DIV 8 : POKE_W b+2,offset+8 : POKE b+4,NL%
380 FOR i=5 TO NL%+4 : POKE i+b,CODE(name$(i-4)) && -33
390 b=b+offset : POKE_L b,0 : POKE_W b+4,0 : b=b+6 : base=b
400 FOR lnum=760 TO 890 STEP 10 : DATASTORE lnum
410 IF base+254<>b : RECHP buffer : PRINT " DATA length error!" : STOP
420 POKE_L base+124,file_data : POKE_L base+130,file_length
430 LBYTES source$,b :REMark Load task to end of code
440 flag=FTEST(destin$) : size=file_length+b-buffer
450 IF flag=0
460   PRINT destin$!'already exists - Delete (Y/N) ? ';
470   k$=INKEY$(#1,-1) : PRINT k$
480   IF k$=='y' THEN DELETE destin$ : flag=-7 : ELSE flag=-8
490 END IF
500 IF flag=-7
510   SBYTES destin$,buffer,size
520   PRINT \\name$!"keyword code has been created in"!destin$
530 ELSE
540   PRINT ' Error writing'!destin$ : REPORT #1,flag
550 END IF
560 RECHP buffer : STOP
570 :
580 DEFine PROCedure DATASTORE(line%)
590 LOCal i, x, words%, key%, checksum%
600 RESTORE line% :REMark Interpreters only (initial checking)
610 key%=3 : READ checksum%, words%
620 FOR i=0 TO words% * 2 -2 STEP 2
630   READ x : POKE_W i+b,x
640   IF x>32767 THEN x=x-65536
650   key%=key% ^^ x ^^ i
660 END FOR i
670 IF key%<>checksum%
680   PRINT "Error in DATA line"!line% : RECHP buffer : STOP
690 END IF
700 b=b + words% * 2
710 END DEFine DATASTORE
720 :
730 REMark BASIC Initialisation code
740 DATA 14403,5,17402,8,13432,272,20178
750 REMark Resident procedure code
760 DATA 1777,9,28922,20085,32256,31232,48075,26478,2101,4,59641
770 DATA 9730,9,26384,2245,31,18997,59640,26118,20877,48075,26454
780 DATA 8803,9,18997,59641,27422,10315,18413,65528,13432,278,20114
790 DATA 21657,9,26314,14897,59392,26114,21641,47563,26422,10827,9804
800 DATA -18292,9,13432,280,20114,26290,15875,10313,12337,59394,27558
810 DATA -28691,10,49404,40,53422,48,45230,52,27800,8246,2048,27538
820 DATA -31025,11,9008,59392,22665,21315,26332,8780,9788,0,0,9276,0
830 DATA -28043,9,2,29439,37833,28673,20033,19072,26218,18928,10248
840 DATA -3989,10,18420,14584,59530,8776,17914,94,8922,8922,8922,8922
850 DATA 16476,8,20938,65526,12295,53312,19013,26116,16995,24582
860 DATA -17263,9,22085,57933,53317,38592,38592,47052,25894,10315,8814
870 DATA 26696,9,88,24582,14065,59392,21641,20936,65528,19015,26122
880 DATA 18685,9,32257,9326,48,10550,43048,14599,8524,65524,29712
890 DATA 20555,7,30463,19077,27138,30208,28682,20033,20085

((END))
```

DBQL

In the fourth part of his relational database, Tom Ashcroft implements multiple file opening.

```
1000 start
1010 REMark ************************ START
1020 DEFine PROCedure start
1030 MODE 4
1040 OPEN #3,ser1
1050 OPEN #4,con_480x30a10x0
1060 WINDOW 480,206,10,30
1070 WINDOW#0,480,20,10,236
1080 nemo=1:chan=1:alias=1:databases=0:lprint=0
1090 ann=1:afields=1:achars=1:adeleted=0:aindices=0:ac=1:aresave=0:
aoldfile=0:apac=0:af$="":aactive=1:aselected=0
1100 bnn=1:bfields=1:bchars=1:bdeleted=0:bindices=0:bc=1:bresave=0:
boldfile=0:bpac=0:bf$="":bactive=1:bselected=0
1110 cnn=1:cfields=1:cchars=1:cdeleted=0:cindices=0:cc=1:cresave=0:
coldfile=0:cpac=0:cf$="":cactive=1:cselected=0
1120 PAPER 0:PAPER #0,0:PAPER#4,0:INK 7:INK #0,7:INK#4,4:CLS:CLS#0
1130 menu
1140 END DEFine start
1150 REMark ************************ MENU
1160 DEFine PROCedure menu
1170 CLS#4
1180 PRINT#4,"CReate    OPen    ENter   Goto   Back    Next    FInd
MOre   LOokup   AMend   PRinter    DELete    SElect   DEselect   LIst
USE   INdex    PAck    QUIT"
1190 IF lprint THEN PAPER#4,4:INK#4,7:PRINT#4," Is the printer ready?
":PAPER#4,0:INK#4,4
1200 END DEFine menu
1210 REMark ************************ ENTER
1220 DEFine PROCedure en
1230 REPeat entrygroup
1240 SELect ON alias
1250    =1:enter a$,ann,afields,aresave,achars,aindices,andx$, aactive
1260    =2:enter b$,bnn,bfields,bresave,bchars,bindices,bndx$, bactive
1270    =3:enter c$,cnn,cfields,cresave,cchars,cindices,cndx$, cactive
1280 END SELect
1290 ssave:lload
1300 IF q$="@" THEN EXIT entrygroup
1310 END REPeat entrygroup
1320 CLS:menu
1330 END DEFine en
1340 DEFine PROCedure
enter(x$,nn,fields,resave,chars,indices,ndx$,active)
1350 CLS:CLS#4:resave=1
1360 PRINT#4," Type information field by field      Key ENTER at the
end of each field"\"                              Shift/ESC to quit"
1370 REPeat entryloop
1380  nn=nn+1:CLS:PRINT"Record no ";nn
1390  FOR j=1 TO fields STEP 19
1400   FOR k=j TO j+18
1410    IF k>fields THEN EXIT k
1420    PRINT x$(0,k,4 TO LEN(x$(0,k)));TO 10;";"
1430   END FOR k
1440   FOR k=j TO j+18
1450    IF k>fields THEN EXIT k
1460    AT (k-j)+1,12;
1470    IF x$(0,k,3)="N" THEN
1480      q$=intake$:IF q$="" THEN q$="0"
1490    ELSE
1500    IF x$(0,k,3)="D" THEN
1510      q$=checkdate$:IF q$="" THEN q$="000000"
1520      q$=date_rev$(q$)
1530    ELSE
1540      INPUT q$
1550    END IF :END IF
1560      IF q$="@" THEN nn=nn-1:EXIT entryloop
1570      x$(nn,k)=q$
1580   END FOR k
1585   AT 1,0:CLS 2:CLS 3
1590 END FOR j
1600 x$(nn,0)=" "
```

The ability to open more than one file simultaneously is a great step forward in the versatility of a database system. It makes it possible to take information from one file to enter in another, or to combine with data from another file or files to form a composite report. Again, two or more files can be linked by a common field so that each contains information about one set of relationships and can be related to the other. For example, a car hire company might have one database containing information about customers such as name, address, date and duration of hire, method of payment, previous hirings etc., and a second database containing information about the fleet such as make, model, engine capacity, date of purchase, date of next service etc. These two databases might have a field, such as a 'car registration number', which is common to both. One file contains everything relating to the customer and the other everything relating to the car, linked by the common field, hence the name **relational database.** To use such a system it must be possible to manipulate and move between databases with ease.

Two ways

Turning to DBQL, there are two possible ways of implementing multiple files. The easiest is to create A$ as a multidimensional array. DIM a$ (5,10,15,20) would provide for five arrays, each with 10 records consisting of 15 fields, each 20 characters in length. The drawback to this system is that all the arrays are automatically dimensioned to the same size and cannot be varied, and the size would obviously have to accommodate the largest number of records, fields and characters of any of the databases to be loaded, irrespective of the needs of the smaller files. If we had a large file of a thousand records of 20 fields and a small look-up table with 10 records of two fields, both would have to be held in identically dimensioned arrays, with most of the second array consisting of empty strings, and a great waste of memory. It would always be necessary to load the largest file first, to ensure the arrays are made large enough.

The alternative approach is to have a separate array for each database so that each can be dimensioned exactly to the right size. This approach is the one used in commercial database systems and is the one we will use in DBQL. Each array will have a different name – a$, b$, c$ . . . etc – and there will have to be a selection

mechanism to call the array which is to be in use at any one time. Superbasic has the SELect ON . . . End SELect structure for this purpose. In commercial programs each database is usually given an 'alias' name by which it can be addressed and which is different from its file name. Superbasic can only SELect on numbers, unfortunately, so each of our files will be known by a number and this will be stored in the variable *alias*.

We need to arrange the program so that, though several different files are held in memory, the program behaves as if, at any one moment, only one file is present and the others remain unseen. Each file will have its associated variables such as nn, fields, chars, etc and each can have associated index files too. Bearing in mind that the procedures of DBQL are written for an array called a$ how can they work for b$ or c$?

Alias and variables

The solution to these problems is simple in principle. Each array will have its own set of variables, so that a$ has ann, afields, achars etc and b$ jas bnn, bfields, bchars and so on. The number of the file in use is held in alias and the various procedures will each have a new introductory procedure to pass the appropriate variables to the procedure proper (see the *QL User Guide* on passing variables). As an example let's look at the changes to the procedure AM (line 2430 in the listing). Please note that the program has been renumbered.

Calling AM from the keyboard now enters the SELect structure and if the value of alias is, say, 1 then the new procedure AMEND is called and has passed to it the array a$ and all the associated variables listed as parameters in line 2450. If alias = 2 then b$ and its variables would be passed. These are the variables that will be needed by AMEND and by the other procedures which will be called by AMEND as it runs. AMEND itself starts at 2500 and consists of the lines which previously made up the old procedure AM in the flat-file version of DBQL. It lists as formal variables in 2450, and must be listed in the same order. The values of some of these variables, of course, are changed as the procedure runs but the new values are automatically passed back to the actual parameter variables.

You will see in the **listing one** that all of the old procedures have been changed in a similar way except for those that are always called by another procedure. This is because variables passed to a procedure are automatically passed on to any other procedure called by the first one. The short procedures DEL and G are too simple to need subsidiary procedures and operate with the actual parameter variables only. Provision is made for up to three files to be held at once.

Procedure LLOAD is especially com-

```
1610 update_index
1620 IF nn=DIMN(x$)THEN PRINT"Saving to microdrive. Please wait.":EXIT
entryloop
1630 END REPeat entryloop
1640 END DEFine enter
1650 REMark ********************** SEARCH
1660 DEFine PROCedure fi
1670 c=0:found=0:CLS
1680 INPUT\"Enter item for search ";s$
1690 SELect ON alias
1700    =1:ac=0:more a$,ann,afields,ac
1710    =2:bc=0:more b$,bnn,bfields,bc
1720    =3:cc=0:more c$,cnn,cfields,cc
1730 END SELect
1740 END DEFine fi
1750 REMark ************** CONTINUE SEARCH
1760 DEFine PROCedure mo
1770 SELect ON alias
1780    =1:more a$,ann,afields,ac
1790    =2:more b$,bnn,bfields,bc
1800    =3:more c$,cnn,cfields,cc
1810 END SELect
1820 END DEFine mo
1830 REMark ************************ MORE
1840 DEFine PROCedure more(x$,nn,fields,c)
1850 d=c+1:IF d>nn THEN PRINT\ TO 9;:PAPER 2:PRINT" end of file ":PAPER
0:RETurn
1860 instr_sr
1870 IF c=nn+1 THEN :IF found THEN :PRINT "NO FURTHER RECORDS
FOUND":ELSE :PRINT\"NOT FOUND":END IF :RETurn :END IF
1880 display x$,fields,c
1890 END DEFine more
1900 REMark ********************* INSTR_SR
1910 DEFine PROCedure instr_sr
1920 c=nn+1
1930 FOR record=d TO nn
1940   FOR fld= 1 TO fields
1950     IF s$ INSTR x$(record,fld) AND x$(record,0,2)<>"S" THEN
c=record:found=1:EXIT record
1960   END FOR fld
1970 END FOR record
1980 END DEFine instr_sr
1990 REMark ********************** DELETE
2000 DEFine PROCedure del
2010 SELect ON alias
2020   =1:a$(ac,0,1)="D":adeleted=adeleted+1
2030   =2:b$(bc,0,1)="D":bdeleted=bdeleted+1
2040   =3:c$(cc,0,1)="D":cdeleted=cdeleted+1
2050 END SELect
2060 PRINT\ "This record is marked for deletion"
2070 END DEFine del
2080 REMark ********************** PRINTER
2090 DEFine PROCedure pr
2100 lprint=1-lprint:menu
2110 END DEFine pr
2120 REMark ************************ GOTO
2130 DEFine PROCedure g(x)
2140 SELect ON alias
2150    =1:ac=x:IF ac>ann THEN ac=ann:END IF :display a$,afields,ac
2160    =2:bc=x:IF bc>bnn THEN bc=cnn:END IF :display b$,bfields,bc
2170    =3:cc=x:IF cc>cnn THEN cc=cnn:END IF :display c$,cfields,cc
2180 END SELect
2190 END DEFine g
2200 REMark ************************ NEXT
2210 DEFine PROCedure n
2220 SELect ON alias
2230   =1:nxt ac,ann:display a$,afields,ac
2240   =2:nxt bc,bnn:display b$,bfields,bc
2250   =3:nxt cc,cnn:display c$,cfields,cc
2260 END SELect
2270 END DEFine n
2280 DEFine PROCedure nxt(c,nn)
2290 c=c+1:IF c>nn THEN PRINT\ TO 30;"end of file":PAUSE 50:c=nn
2300 END DEFine nxt
2310 REMark ************************ BACK
2320 DEFine PROCedure b
2330 SELect ON alias
2340   =1:back ac,ann:display a$,afields,ac
2350   =2:back bc,bnn:display b$,bfields,bc
2360   =3:back cc,cnn:display c$,cfields,cc
2370 END SELect
2380 END DEFine b
2390 DEFine PROCedure back(c,nn)
2400 c=c-1:IF c<1 THEN PRINT\ TO 30;"start of file":PAUSE 50:c=1
2410 END DEFine back
2420 REMark ********************** AMEND
2430 DEFine PROCedure am
2440 SELect ON alias
2450   =1:amend a$,ann,afields,achars,aresave,adeleted,ac,
aindices,andx$
2460   =2:amend b$,bnn,bfields,bchars,bresave,bdeleted,bc,
bindices,bndx$
```

```
2470    =3:amend c$,cnn,cfields,cchars,cresave,cdeleted,cc,
cindices,cndx$
2480 END SELect
2490 END DEFine am
2500 DEFine PROCedure
amend(x$,nn,fields,chars,resave,deleted,c,indices,ndx$)
2510 CLS:CLS#4
2520 PRINT#4,"                          ENTER leaves item unchanged "
2530 PRINT#4,"                       To change an item, type new version
2540 DIM hold$(fields,chars)
2550 resave=1:reenter=0
2560 FOR fld=1 TO fields
2570    PRINT x$(0,fld,4 TO LEN(x$(0,fld)));TO 11;x$(0,fld,3);":
"&x$(c,fld)
2580 PRINT TO 11;
2590 IF x$(0,fld,3)="N" THEN
2600     q$=intake$
2610 ELSE
2620 IF x$(0,fld,3)="D" THEN
2630     q$=checkdate$
2635     q$=date_rev$(q$)
2640 ELSE
2650     INPUT q$
2660 END IF :END IF
2670    IF q$="" THEN :hold$(fld)=x$(c,fld):ELSE
:hold$(fld)=q$:reenter=reenter+(x$(0,fld,1)="I"):END IF
2680 END FOR fld
2690 IF reenter THEN
2700    x$(c,0)="D":deleted=deleted+1:nn=nn+1
2710    FOR j=1 TO fields:x$(nn,j)=hold$(j)
2720    update_index
2730 ELSE
2740    FOR j=1 TO fields:x$(c,j)=hold$(j)
2750 END IF
2760 IF reenter THEN c=nn
2770 display x$,fields,c:menu
2780    IF nn=DIMN(x$) THEN PRINT"Saving file to microdrive. Please
wait.":ssave:lload
2790 END DEFine amend
2800 REMark ************************ QUIT
2810 DEFine PROCedure quit
2820 FOR j=1 TO databases
2830     alias=j
2840     ssave
2850 END FOR j
2860 ending
2870 END DEFine quit
2880 REMark ************************ SAVE
2890 DEFine PROCedure ssave
2900 SELect ON alias
2910    =1:saver a$,ann,afields,achars,adeleted,apac,aoldfile,af$,
aresave,aindices,andx$
2920    =2:saver b$,bnn,bfields,bchars,bdeleted,bpac,boldfile,bf$,
bresave,bindices,bndx$
2930    =3:saver c$,cnn,cfields,cchars,cdeleted,cpac,coldfile,cf$,
cresave,cindices,cndx$
2940 END SELect
2950 END DEFine ssave
2960 DEFine PROCedure saver(x$,nn,fields,chars,deleted,pac,oldfile,f$
resave,indices,ndx$)
2970 IF NOT resave THEN RETurn
2980 dbsave:indexsave:resave=0
2990 END DEFine saver
3000 DEFine PROCedure dbsave
3010 IF oldfile THEN COPY "mdv1_"&f$ TO "mdv1_"&f$&"_temp":DELETE
"mdv1_"&f$
3020 OPEN_NEW #6,"mdv1_"&f$
3030 PRINT#6,nn-(deleted*(pac=1)):PRINT#6,fields:PRINT#6,chars:
PRINT#6,deleted*(1-pac):PRINT#6,indices
3040 FOR record=0 TO nn
3050 IF x$(record,0,1)="D" AND pac=1 THEN NEXT record
3060 FOR fld=0 TO fields
3070 PRINT#6,x$(record,fld)
3080 END FOR fld:END FOR record
3090 CLOSE#6
3100 DELETE "mdv1_"&f$&"_temp"
3110 END DEFine dbsave
3120 DEFine PROCedure indexsave
3130   FOR j=1 TO fields
3140     IF x$(0,j,1)="I" THEN
3150     IF NOT pac THEN COPY "mdv1_"&x$(0,j,4 TO LEN(x$(0,j)))&"_ind"
TO "mdv1_"&x$(0,j,4 TO LEN(x$(0,j)))&"_ind_temp"
3160     DELETE "mdv1_"&x$(0,j,4 TO LEN(x$(0,j)))&"_ind"
3170     IF pac THEN NEXT j
3180     OPEN_NEW#6, "mdv1_"&x$(0,j,4 TO LEN(x$(0,j)))&"_ind"
3190     FOR k=0 TO nn:PRINT#6, ndx$(x$(0,j,2),k,1):PRINT#6,
ndx$(x$(0,j,2),k,2)
3200     CLOSE#6
3210     DELETE "mdv1_"&x$(0,j,4 TO LEN(x$(0,j)))&"_ind_temp"
3220     END IF
3230   END FOR j
3240 END DEFine indexsave
```

plicated because of the fact that an array can only be dimensioned as an actual variable and not in the guise of a formal parameter. This means that LLOAD has to be split into two procedures. SET_ARRAY opens the file and reads in the numerical variables and the program then returns to the calling line to have the main array dimensioned before passing the variables on to INLOAD, which loads the data into the array. If any index files are present the appropriate ndx$ has to be dimensioned at the same time as the database array so the value of *indices* has to be known at this point. Indices is therefore now saved in the file with the other variables, although the older lines which derived indices from the field names have been left as a check. To make database files compatible with both versions of DBQL it is recommended that previous versions be modified to save indices to microdrive.

The procedure START has been changed to initialise the new database variables so that they can be used as passing parameters. Two new global variables have also been added – alias, which has been explained, and *databases*, which stores the number of files currently in memory. It begins with a value of 0 and is incremented whenever a database is loaded or created. Alias begins with a value of 1 and to load the first file call OP in the usual way. To load another file, set alias to another value, up to three, and call OP again. The value of alias is changed by a new procedure USE, passing the required value from the keyboard as USE 2 or USE 3 etc. another way is just to type 'alias=x' <ENTER>, where x is the new value. If you wish to replace one file with another, set alias to the required number and OPen the new file, which will automatically replace the existing one.

As listed, the maximum number of files that can be loaded at once is three. Commercial databases usually allow more files eg *dBase IV* takes up to 10 and some users need still more. The number of files in DBQL can be increased quite easily and the only real limit is the amount of memory available. To add a fourth file, which would be called d$, it would be necessary to add a new line to each of the SElect structures:

=4: [procedure name] d$,dnn,dfields. . . etc,

and list all the d- variables corresponding to the a-, b- and c- variables in the existing lines. A new line would similarly be required in START to initialise the new variables, and the maximum value allowed by USE should be increased. This process could be repeated as often as desired, using the next letter of the alphabet each time as a name for the new array.

The program has been completely renumbered to accommodate the new lines. Line numbers in DBQL are only needed for the programmer to find the way around and are not used by the program itself. A

number of other additions have been made.

SELEKT has been altered to enable 'greater than ' and 'less than' criteria to be used in selections. The new SELEKT, after printing instructions in #4 and displaying the list of field numbers and names, requests an entry in the form of an unbroken string consisting of field number, operator and selection item. The string is then analysed by lines 4270 to 4340. The loop in 4270 to 4290 examines each character in the string until it finds a non-numeric character. If the first character is non-numeric then the string has been incorrectly entered and 4300 prints an error message and loops back to the input.

Line 4310 takes the characters before the first non-numeric character to be the field number and assigns this to *fldno*. The non-numeric character itself is taken to be the operator and assigned to operator$, while the rest of the string is taken to be the selection item and assigned to selct$. Line 4320 checks the validity of the operator and fieldnumber and if one of these is wrong the repeat loop returns to the input with an error message.

If all is well here, the selection loop starting at 4370 is entered, after a numerical value has been obtained from the ASCII code of the operator character in 4350. The loop examines the specified field of each record in the database using a SELect structure to choose the appropriate operator and if a positive result is found it increments the counter variable *selected*. Otherwise a 'S' is attached to the end of field 0 of the record, which is a change from the previous arrangement (see below). At the end of the loop the number of records selected is displayed as before and the same options of displaying or saving the selected records are offered.

The deselection procedure DES has also been changed. Experience has shown that it is often useful to be able to cancel only the last stage of a multiple selection to allow a new selection to be based on a previous one. For instance, a mail order firm, having selected the names of male customers aged under 30, say, might want to select those living in Birmingham and then those living in Brighton or Newcastle in turn, without having to cancel the whole selection and begin anew each time. This is done by adding an 'S' to field 0 whenever a record is selected out, so that a record which has failed three selections would have SSS in its 0 field (after the first character which is reserved for 'D') while one which has only failed the last selection would only have one S.

To go back to the state existing before the latest selection therefore, the last S has to be removed from each record containing Ss, and this option is now offered by DESLCT. Line 4710 offers the choice of complete cancellation as before by inputting a C (upper or lower case) or of cancelling only the last selection by entering S. If C is entered, the loop in 4370 removes all Ss by setting field 0 in each record equal to

```
3250 REMark ************************ OPEN
3260 DEFine PROCedure op
3270 INPUT "Enter file name: ";q$
3280 SELect ON alias
3290   =1:af$=q$:lload:legend af$,ann,aindices
3300   =2:bf$=q$:lload:legend bf$,bnn,bindices
3310   =3:cf$=q$:lload:legend cf$,cnn,cindices
3320 END SELect
3330 databases=databases+1-(databases=3)
3340 END DEFine op
3350 DEFine PROCedure lload
3360 SELect ON alias
3370   =1:set_array af$,ann,afields,achars,adeleted,aindices:DIM
a$(ann+10,afields,achars):DIM andx$(aindices,ann+10,2,achars):inload
a$,ann,afields,aindices,apac,andx$,aoldfile
3380   =2:set_array bf$,bnn,bfields,bchars,bdeleted,bindices:DIM
b$(bnn+10,bfields,bchars):DIM bndx$(bindices,bnn+10,2,bchars):inload
b$,bnn,bfields,bindices,bpac,bndx$,boldfile
3390   =3:set_array cf$,cnn,cfields,cchars,cdeleted,cindices:DIM
c$(cnn+10,cfields,cchars):DIM cndx$(cindices,cnn+10,2,cchars):inload
c$,cnn,cfields,cindices,cpac,cndx$,coldfile
3400 END SELect
3410 END DEFine lload
3420 DEFine PROCedure legend(f$,nn,indices)
3430 CLS:AT 10,7:PRINT"The file '"&f$&"' is open":PRINT\ TO 13;nn;"
records"
3440 IF indices THEN PRINT\ TO 13;indices;" index file(s)"
3450 END DEFine legend
3460 REMark ******************** SET ARRAY
3470 DEFine PROCedure set_array(f$,nn,fields,chars,deleted,indices)
3480 OPEN_IN #6,"mdv1_"&f$
3490 INPUT#6,nn:INPUT#6,fields:INPUT#6,chars:INPUT#6,deleted:
INPUT#6,indices
3500 END DEFine set_array
3510 REMark ********************** INLOAD
3520 DEFine PROCedure inload(x$,nn,fields,indices,pac,xndx$,oldfile)
3530 FOR record=0 TO nn
3540 FOR fld=0 TO fields
3550 INPUT#6,x$(record,fld)
3560 END FOR fld
3570 END FOR record
3580 CLOSE #6:oldfile=1
3590 IF pac THEN RETurn
3600 indices=0:FOR j=1 TO fields:IF x$(0,j,1)="I" THEN
indices=indices+1:x$(0,j,2)=indices
3610 IF NOT indices THEN RETurn
3620 FOR j=1 TO fields
3630   IF x$(0,j,1)="I" THEN
3640     OPEN_IN #6, "mdv1_"&x$(0,j,4 TO LEN(x$(0,j)))&"_ind"
3650     FOR k=1 TO nn:INPUT#6, xndx$(x$(0,j,2),k,1):INPUT#6,
xndx$(x$(0,j,2),k,2)
3660     CLOSE#6
3670   END IF
3680 END FOR j
3690 END DEFine inload
3700 REMark ********************** CREATE
3710 DEFine PROCedure cr
3720 IF alias-databases<1 THEN alias=alias+1-(alias=3)
3730 CLS
3740 INPUT"Please enter file name: ";f$
3750 x$="":fields=0
3760 CLS#4: PRINT#4,"Enter field names in order, up to 8 characters in
length."\"Enter Shift/ESC to quit."
3770 PRINT"    Field name          Character, Number or Date"
3780 REPeat fname
3790 INPUT\ fields+1;" : "; field$;
3800 IF field$="©" THEN EXIT fname
3810 INPUT TO 28;type$;:type$=upper$(type$(1))
3820 IF NOT type$ INSTR "CND" THEN INPUT" Invalid type. Only C,N or D
";type$
3830 IF LEN(field$)>8 THEN field$=field$(1 TO 8)
3840 field$="  "&type$&field$
3850 x$=x$&field$&",":fields=fields+1
3860 END REPeat fname
3870 INPUT\"How many characters in the longest field?"!chars
3880 IF chars<11 THEN chars=11
3890 DIM z$(0,fields,chars)
3900 FOR j=1 TO fields
3910   slice=," INSTR x$:z$(0,j)=x$(1 TO slice-1)
3920   IF slice<LEN(x$) THEN x$=x$(slice+1 TO)
3930 END FOR j
3940 nn=0:deleted=0:indices=0
3950 OPEN_NEW#6,"mdv1_"&f$
3960 PRINT#6,nn:PRINT#6,fields:PRINT#6,chars:PRINT#6,deleted:
PRINT#6,indices
3970 FOR j=0 TO fields:PRINT#6,z$(0,j)
3980 CLOSE #6
3990 databases=databases+1-(databases=3)
4000 CLS:CLS#0:INPUT\"The file '"&f$&"' is ready. Enter records
now?(Y/N) ";q$
4010 IF q$=="Y" THEN
4020   SELect ON alias
4030     =1:af$=f$:lload:en
```

```
4040     =2:bf$=f$:lload:en
4050     =3:cf$=f$:lload:en
4060   END SELect
4070 END IF
4075 CLS:menu
4080 END DEFine cr
4090 REMark ********************** ENDING
4100 DEFine PROCedure ending
4110 CLS:CLS#0:AT 10,33:PRINT"FILES CLOSED"\\ TO 27;"To use again,
enter RUN."
4120 END DEFine ending
4130 REMark ********************** SELECT
4140 DEFine PROCedure se
4150 SELect ON alias
4160     =1:selekt a$,ann,afields,aselected,ac
4170     =2:selekt b$,bnn,bfields,bselected,bc
4180     =3:selekt c$,cnn,cfields,cselected,cc
4190 END SELect
4200 END DEFine se
4210 REMark ********************** SELEKT
4220 DEFine PROCedure selekt (x$,nn,fields,selected,c)
4230 AT#4,0,0:PRINT#4,"Enter field number, operator(=,< or >) and
selection item as a continuous stringof characters e.g. 10=bingo
"
4240 CLS:FOR j=1 TO fields:PRINT j,x$(0,j,4 TO LEN(x$(0,j)))
4250 REPeat selection
4260 INPUT q$
4270 FOR j=1 TO LEN(q$)
4280     IF NOT (CODE(q$(j))<58 AND CODE(q$(j))>47) THEN EXIT j
4290 END FOR j
4300 IF j=1 THEN PRINT "Invalid entry. Refer to instructions":NEXT
selection
4310 fldno$=q$(1 TO j-1):fldno=fldno$:operator$=q$(j):selct$=q$(j+1 TO)
4320 IF operator$ INSTR "=<>" AND fldno$<= fields THEN EXIT selection
4330 PRINT"Invalid entry. Refer to instructions"
4340 END REPeat selection
4350 operator= CODE(operator$)
4360 selected=0
4370 FOR j=1 TO nn
4380     SELect ON operator
4390         =61:IF selct$ INSTR x$(j,fldno) AND x$(j,0,2)<>"S" THEN
selected=selected+1:ELSE :x$(j,0)=x$(j,0,1 TO LEN(x$(j,0)))&"S":END IF
4400         =60:IF x$(j,fldno)<selct$ AND x$(j,0,2)<>"S" THEN
selected=selected+1:ELSE :x$(j,0)=x$(j,0,1 TO LEN(x$(j,0)))&"S":END IF
4410         =62:IF x$(j,fldno)>selct$ AND x$(j,0,2)<>"S" THEN
selected=selected+1:ELSE :x$(j,0)=x$(j,0,1 TO LEN(x$(j,0)))&"S":END IF
4420     END SELect
4430 END FOR j
4440 PRINT selected;" records selected"
4450 IF NOT selected THEN menu:RETurn
4460 INPUT"Enter S to save selected records to microdrive"\"       L to
list them on screen"\"Press ENTER to return to menu ";q$
4470 IF q$="S" OR q$="s" THEN
4480     INPUT "Enter file name ";subfile$
4490     OPEN_NEW#6, "mdv1_"&subfile$
4500     PRINT#6,selected: PRINT#6,fields: PRINT#6,chars: PRINT#6,0:
PRINT#6,0
4510     FOR j=0 TO nn
4520     IF x$(j,0,2)="S" THEN NEXT j
4530     FOR k=0 TO fields:PRINT#6,x$(j,k)
4540     END FOR j:CLOSE#6
4550 END IF
4560 IF q$="l" OR q$="L" THEN
4570     CLS:FOR k=1 TO nn:IF x$(k,0,2)<>"S" THEN c=k:b_s_display
4580 END IF
4590 menu
4600 END DEFine selekt
4610 REMark ********************** DESELECT
4620 DEFine PROCedure des
4630 SELect ON alias
4640     =1:deslct a$,ann,aselected
4650     =2:deslct b$,bnn,bselected
4660     =3:deslct c$,cnn,cselected
4670 END SELect
4680 END DEFine des
4690 REMark ********************** DESLCT
4700 DEFine PROCedure deslct(x$, nn, selected)
4710 PRINT\ "Enter C to cancel all selections"\"       S to step back
one stage":INPUT opt$
4720 IF opt$="C" OR opt$="c" THEN
4730     FOR j=1 TO nn:x$(j,0)=x$(j,0,1)
4740     selected=0
4750     CLS:PRINT\"All selections cancelled"
4760 ELSE
4770 IF opt$="S" OR opt$="s" THEN
4780     selected=0
4790     FOR j=1 TO nn
4800     IF LEN(x$(j,0))>1 THEN x$(j,0)=x$(j,0,1 TO LEN(x$(j,0))-1)
4810     selected=selected+(LEN(x$(j,0))=1)
4820     END FOR j
4830     IF selected=nn THEN PRINT\"All selections now cancelled":ELSE
:PRINT\ "Last selection cancelled"\selected;" records now selected"
4840 ELSE
4850 PRINT\"Deselect abandoned"
4860 END IF :END IF
4870 END DEFine deslct
4880 REMark **********
4890 DEFine PROCedure li
4900 SELect ON alias
4910     =1:show a$,ann,afields,
4920     =2:show b$,bnn,bfields,
4930     =3:show c$,cnn,cfields,
4940 END SELect
4950 END DEFine li
4960 REMark **********
4970 DEFine PROCedure show(x$,
4980 IF NOT selected THEN PRINTNO
4990 CLS:FOR k=1 TO nn:IF x$(k,
5000 END DEFine show
5010 REMark **********
5020 DEFine PROCedure display(
5030 IF chan=1 THEN CLS
5040 PRINT"record ";c;:IF x$(c,
:PRINT:PRINT:END IF
5050 FOR fld=1 TO fields
5060     IF NOT nemo THEN PRINT
11;": ",
5065     IF x$(0,fld,3)="D" THEN
date_rev$(temp$):NEXT fld
5070     PRINT#chan," "&x$(c,fld
5080 END FOR fld
5090 IF lprint AND chan=1 THEN
5100 chan=1
5110 END DEFine display
5120 :
5130 REMark **********
5140 DEFine PROCedure in
5150 SELect ON alias
5160     =1:index a$,afields,
aoldfile,af$
5170     =2:index b$,bfields,
boldfile,bf$
5180     =3:index c$,cfields,
coldfile,cf$
5190 END SELect
5200 END DEFine in
5210 DEFine PROCedure index(x$
deleted,oldfile,f$)
5220 DIM ndx$(1,nn,2,chars):
5230 IF NOT pac THEN
5240     CLS:FOR j=1 TO fields:
5250     PRINT"Enter field numb
fldno THEN PRINT"Terminating":
5260     PRINT"Creating index f
LEN(x$(0,fldno)))&"_ind"
5270 END IF
5280 x$(0,fldno,1)="1"
5290 FOR k=1 TO nn:ndx$(
5300 DIM comp$(2,chars)
5310 FOR item=2 TO nn
5320 in_sort
5330 END FOR item
5340 indname$=x$(0,fldno,
5350 DELETE "mdv1_"&indname$
5360 OPEN_NEW#5,"mdv1_"&
5370 FOR k=1 TO nn:PRINT#5,
5380 CLOSE#5
5390 indices=indices+1
5400 resave=1
5410 dbsave
5420 END DEFine index
5430 REMark **********
5440 DEFine PROCedure in_sort
5450     p=item
5460     comp$(1)=ndx$(
5470     ndx$(active,0,1)=
5480     REPeat compare
5490       IF comp$(1)>=
5500
ndx$(active,p,1)=ndx$(
5510       p=p-1
5520     END REPeat compare
5530     ndx$(active,p,1)=
5540 END DEFine in_sort
5550 REMark **********
5560 DEFine PROCedure lo
5570 CLS:INPUT\"Please
5580 SELect ON alias
5590     =1:b_search a$,
5600     =2:b_search b$,
5610     =3:b_search c$,
5620 END SELect
5630 END DEFine lo
5640 DEFine PROCedure b_
```

```
*** LIST

elected,ac
elected,bc
elected,cc

******* SHOW
fields,selected,c)
NO SELECTION HAS BEEN MADE":RETurn
2)<>"S" THEN c=k:b_s_display

*** DISPLAY
fields,c)

1)="D" THEN PRINT"   ( D )":PRINT:ELSE


an,x$(0,fld,4 TO LEN(x$(0,fld))):TO

emp$=x$(c,fld):PRINT#chan,


an=3:display x$,fields,c


* INDEX FILE

s,ann,apac,aresave,aindices,adeleted,

s,bnn,bpac,bresave,bindices,bdeleted,

s,cnn,cpac,cresave,cindices,cdeleted,


fields,chars,nn,pac,resave,indices,

e =1

NT j,x$(0,j,4 TO)
":fldno=getnumber$(0,fields) :IF NOT
urn
":x$(0,fldno,4 TO


k)=x$(k,fldno):ndx$(active,k,2)=k


dx$(0,fldno))&" ind"

ctive,k,1):PRINT#5,ndx$(active,k,2)


* IN SORT

:comp$(2)=ndx$(active,p,2)
:ndx$(active,0,2)=comp$(2)

p,p-1,1) THEN EXIT compare

ndx$(active,p,2)=ndx$(active,p-1,2)

ndx$(active,p,2)=comp$(2)

RY SEARCH

e look for: ";s$

n,ac,active;andx$,aindices
n,bc,active,bndx$,bindices
n,cc,active,cndx$,cindices

fields,nn,c,active,ndx$,indices)
```

```
5550 REMark ***************** BINARY SEARCH
5560 DEFine PROCedure lo
5570 CLS:INPUT\"Please enter item to look for: ";s$
5580 SELect ON alias
5590     =1:b_search a$,afields,ann,ac,aactive,andx$,aindices
5600     =2:b_search b$,bfields,bnn,bc,bactive,bndx$,bindices
5610     =3:b_search c$,cfields,cnn,cc,cactive,cndx$,cindices
5620 END SELect
5630 END DEFine lo
5640 DEFine PROCedure b_search(x$,fields,nn,c,active,ndx$,indices)
5650 FOR j=1 TO fields:IF x$(0,j,1)="I" THEN PRINT x$(0,j,2),x$(0,j,4
TO)
5660 PRINT\"Current field for search is ";active\"Press ENTER to
confirm "\"or enter another number:  ";: q$=getnumber$(0,indices): IF
q$<>"0" THEN active=q$
5670 FOR sfld= 1 TO fields:IF x$(0,sfld,1 TO 2)="I"&active THEN EXIT
sfld
5680 z=bin_s(nn,s$,ndx$)
5690 c=ndx$(active,z,2)
5700 IF x$(c,sfld,1 TO LEN(s$))<>s$ THEN PRINT\"NOT FOUND ":RETurn
5710 CLS
5720 updown ndx$,z
5730 END DEFine b_search
5740 REMark ***************** FN BIN SEARCH
5750 DEFine FuNction bin_s(n,x$,y$)
5760 LOCal s
5770 power=INT(LN(n-1)/LN(2))
5780 s=2^power
5790 FOR k=power-1 TO 0 STEP -1
5800 s=s+(2^k)*(x$>y$(active,s,1,1 TO
LEN(x$)))-(2^k)*(x$<y$(active,s,1,1 TO LEN(x$)))
5810 IF s>n THEN s=n
5820 IF s<1 THEN s=1
5830 END FOR k
5840 RETurn s
5850 END DEFine bin_s
5860 REMark ************************ UPDOWN
5870 DEFine PROCedure updown(y$,s)
5880 REPeat upward
5890 s=s-1
5900 IF s=0 THEN EXIT upward
5910 c=y$(active,s,2)
5920 IF x$(c,sfld,1 TO LEN(s$))<>s$ THEN EXIT upward
5930 END REPeat upward
5940 REPeat downward
5950 s=s+1
5960 c=y$(active,s,2)
5970 IF x$(c,sfld,1 TO LEN(s$))<>s$ THEN EXIT downward
5980 b_s_display
5990 IF s=nn THEN EXIT downward
6000 END REPeat downward
6010 END DEFine updown
6020  REMark ***************** B S DISPLAY
6030 DEFine PROCedure b_s_display
6040 PRINT#chan, "# ";c&"  ";
6042 FOR j=1 to fields
6044  IF x$(0,j,3)="D" THEN temp$=x$(c,j):PRINT#chan, date_rev$(temp$)
: NEXT j
6046  PRINT#chan, x$(c,j)&" ";
6048 END FOR j
6049 PRINT#chan
6050 IF lprint AND chan=1 THEN chan=3:b_s_display
6060 chan=1
6070 END DEFine b_s_display
6080 REMark ***************** UPDATE_INDEX
6090 DEFine PROCedure update_index
6100 IF NOT indices THEN RETurn
6110 DIM comp$(2,chars)
6120 FOR j=1 TO fields
6130   IF x$(0,j,1)="I" THEN
6140     active=x$(0,j,2)
6150     ndx$(active,nn,1)=x$(nn,j):ndx$(active,nn,2)=nn:item=nn
6160     in_sort
6170   END IF
6180 END FOR j
6190 END DEFine update_index
6200 REMark ************************ PACK
6210 DEFine PROCedure pa
6220 SELect ON alias
6230     =1:pak a$,afields,achars,ann,adeleted,aindices,apac,aresave
6240     =2:pak b$,bfields,bchars,bnn,bdeleted,bindices,bpac,bresave
6250     =3:pak c$,cfields,cchars,cnn,cdeleted,cindices,cpac,cresave
6260 END SELect
6270 END DEFine pa
6280 DEFine PROCedure
pak(x$,fields,chars,nn,deleted,indices,pac,resave)
6290 pac=1:resave=1
6300 ssave
6310 lload
6320 CLS:PRINT\"Rebuilding the index files.  Please wait."
6330 FOR j=1 TO fields
6340     IF x$(0,j,1)="I" THEN
6350       fldno=j
6360       PRINT\x$(0,j,4 TO LEN(x$(0,j)))&" index"
6370       in
6380     END IF
6390 END FOR j
6400 pac=0:resave=0
6410 lload
6420 END DEFine pa
```

```
6430 REMark ************************** USE
6440 DEFine PROCedure use(x)
6450 IF x>3 THEN x=3
6460 alias=x
6470 END DEFine use
6480 REMark ********************** DATE_REVERSE
6490 DEFine FuNction date_rev$ (x$)
6500 slice1$=x$(1 TO 2):slice2$=x$(3 TO 4):slice3$=x$(5 TO 6)
6510 x$=slice3$&slice2$&slice1$
6520 RETurn x$
6530 END DEFine date_rev$
```

its first character only. If option S is entered, field 0 of each record is shortened by removing the last character, provided its length is already more than one. A field of only one character length cannot contain any Ss and is left unchanged. Appropriate messages are printed to screen with each option. Note also that 1600 in ENTER now inserts only one blank character in field 0 in a new record.

If dates are to be selected on using > or < operators then the previous format of date strings is unsuitable, since years are more important than days, and the order of day, month and year needs to be reversed. This is done by a new function DATE_REV$ (line 6490 onwards), which slices the date string into three separate sections for day, month and year and then recombines them in reverse order. The function works equally well for restoring a reversed date to normal format for printing. It is recommended that all dates are stored in reverse form and 1520 reverses a newly entered date in ENTER, so that keyboard entry remains as before, while 2635 does the same for AMEND. Line 5065 prints a reversed (ie normal) date in DISPLAY and line 6044 in B_S_DISPLAY.

CReate now offers the option of entering data immediately into a newly created database. If this option is accepted, the empty file is loaded according to the current setting of *alias* and EN is called. QUIT has been altered to check all the databases in memory and call SSAVE for each in turn.

The changes in SSAVE have made it impossible to address DBSAVE directly and PA(ck) now has to call SSAVE. It is therefore necessary to set and cancel the flag *resave* at the same time as *pac*. These statements have been added to PA.

Facilities for printing output on paper have so far been limited and the system has been radically revised. PR now toggles the value of a global flag *print* which is initially set to 0 by START. DISPLAY has been changed so that when 1 print is set and chan = 1 line 5090 sets chan to 3 and then calls DISPLAY itself – a form of looping – and on the second time round, the output goes to channel 3, the printer. With chan set at 3, line 5090 is skipped on the second pass, (avoiding a perpetual loop!) and 5100 resets chan to 1. The same system has been installed in B_S_ DISPLAY so that results of selections and binary searches can now be printed out. The effect is that with print set, any data printed to screen is echoed to the printer. A new line in MENU now prints a message 'Is the printer ready?' in channel 4 whenever 1 print is set. If the printer is not ready the program will 'hang', waiting for the printer, and this condition could be mistaken for a 'crash'.

Two modifications have been made to ENter. The previous method of saving and reloading the file after entering 10 records no longer works with the new SELect structures and instead the whole procedure has been enclosed in a repeat loop — *entryloop* — which has the same effect. Secondly the display of field names has been changed by the addition of the nested FOR — NEXT loops j and k in 1390 and 1400 so that all the field names are displayed at once, instead of one by one, before input begins with the first field. If the database has more than 19 fields (the number of lines in window #1) the first 19 are displayed, followed by the remainder when the first 19 have been filled.

A full listing of the amended program is included in this article except for the validation functions INTAKE$, CHECKDATE$, UPPER$, LOWER$ and GETNUMBER$. These are exactly the same as in last month's listing, except for new line numbers, and should be added at the end of this listing.

Next month I will deal with programming the database and tailoring it for specific applications.

aCcLcesBs

BELGIUM

Club Sinclair BruQsL (Belgium) Contact: Jaques Tasset, Aarlenstraat 104, 1040 Brussels, Belgium

SWEDEN

International QL Conference bulletin board system (Swedish and English). Contact: Michael Cronsten, System Operator, Jamten-TCL, S Soere 1073, 83030 Lit, Sweden.

USA

New England Sinclair QL User Group (USA) Membership Secretary: Sherm Waterman, PO Box 8763, Boston, MA 02114 8763, USA. Magazine: *NESQLUG News*. Editor: Peter Hale, 195 Central Ave., Chelsea, MA 02150, USA.

NORWAY

Norweigan All Sinclair Association (NASA) Contact: P Monstad, NASA, N-5580 Oelen, Norway. Magazine: *Sinclair Magazine*.

ITALY

Qitaly Club Chairman: Roberto Orlandi, Via Brescia 26, 25039 Travagliato (BS), Italy. Tel. (local) +39 30 6863311. Magazine: *Qitaly Magazine*. Editor: Dr Eros Forenzi, Via Valeriana 44, 23010 Berbenno (SO), Italy. Tel. (local) +39 342 492323.

TURKEY

QL Qlub (Turkey). Contact: Bulent Artuz, Prof. Sitesi B/1 D/5, Etiler 80600, Istanbul, Turkey.

ENGLAND

Quanta (UK) Membership Secretary, Bill Newell, 213 Manor Road, Benfleet, Essex SS7 4JD. Magazine: *Quanta*. Editor: Bill Fuggle, 20 Widnes Avenue, Selly Oak, Birmingham B29 6QE.

Bristol sub-group: Roy Brereton, 94 Teignmouth Rd, Clevedon, Avon BS21 6DR.

Essex sub-group: Bob Fingell, 22 Paley Gardens, Loughton, Essex IG10 2AN.

London sub-group: Jeremy Davis, 6 Elmcroft Crescent, Harrow, Middlesex HA2 6HN.

Northern Ireland: Billy Turkington, Fairyhill, Rostrevor, Newry, Co., Down BT34 3BB.

Qubbesoft PD library. Contact: Ron Dunnett, 38 Brunwin Rd, Rayne, Braintree, Essex CM7 5BU.

SCOTLAND

Scottish QL Users Group Contact: Alan Pemberton, 65 Lingerwood Rd., Newtongrange, Midlothian EH22 4QQ. Newsletter.

HOLLAND

Sin_QL_Air (Netherlands) Membership Secretary: Bob Visser, Snelrewaard 6, 2904 SN Capelle, a/d Ijssel, Netherlands. Magazine: *Quasar*. Editor: C H M Biemans, Elzenstraat 5, 5461 CL Veghel, Netherlands.

GERMANY

Sinclair QL User Club eV (Germany) Foreign Contact: Franz Herrmann, Talstrasse 21, d-W5460 Ochenfels, West Germany. Magazine: *Quasar*.